“拍照赚钱”任务优化定价

摘要

本文针对“拍照赚钱”的任务定价问题，使用回归分析、卡方检验、xgboost、几何覆盖图和非线性整数规划等方法，分别建立了概率定价模型、xgboost 预测模型、非线性整数规划模型和打包定价模型，综合运用了 Python、R 和 Excel 等软件进行数据分析和编程，得到了比较准确的任务完成度评价体系，创新性地针对本问题中的非线性整数规划优化问题提出了效果很好的启发式求解算法，提出了更广泛适用的 PhaseI-PhaseII 优化定价体系，解决了新项目任务的最优定价问题。

针对问题一，在对数据充分分析后，我们将任务的位置信息转化为其与用户群体的交互关系信息，并利用回归分析发现其解释效果很好，能够比较真实地反应项目的定价规律。对于任务未完成原因的分析，我们提取了价格和许多不同距离范围内的用户密度、用户信誉均值信息，去除共线性后，用 R 语言精细调参训练了 xgboost 模型用于预测。最终通过重要性矩阵的分析得到了与完成与否息息相关的特征，进行未完成原因的分析。

针对问题二，由于价格提升会使任务完成率增加，但实际情况中一个项目的预算有限，因此首先引入预算这一概念，并借由 xgboost 的预测结果建立了以任务完成数为目标函数的非线性整数规划问题；其次，我们创新性地提出了针对该问题的一个启发式算法，在本问题中优化效果非常好，不仅大幅提升了任务完成率，更节省了预算。

针对问题三，我们首先考虑市场实际分配情况，对用户进行优先度排序；并以此用几何覆盖思想遍历地寻找用户周围任务集中区域，进行打包；最终，在原有定价体系中给出了打包任务等价任务，并以此给出打包任务的定价，并用同样的方法分析了其效果。

针对问题四，我们首先分析了新任务定价方案与原有模型间的关系；其次，提出了将问题一中概率定价模型和问题二中的优化定价模型结合的方法，得到了 PhaseI-PhaseII 优化定价体系。在通过 xgboost 模型进行任务完成率表现时发现其表现很好，在限制了预算的情况下，比 PhaseI 用概率定价模型的任务完成率高了 30%。最后，我们指出这是一个针对这一类定价问题的一个普适定价体系。

在模型评价中，也认识到我们的模型具有创新性地结合了许多不同领域的或前沿或经典的方法，为这个新型定价问题提出了丰富而有效的一系列定价方法和分析评价方法，而且其效率较高，效果评价比较准确，具有一定的鲁棒性；相应地，我们的方法也有许多不足，比如在打包定价模型中部分参数是人为设定的；在特征选择的时候只选取了 41 个交互信息变量进行特征工程，不能说做的非常精细；而且 xgboost 本身是一个随机算法，所以每次结果会由于数据量的有限而产生小幅波动。虽然这些不足一部分是由于有限的时间和较少的原始数据而不可避免产生的，但是我们对于数据处理和模型建立也有许多不熟练和需要调整的地方，因此我们今后的改进方向包括收集更多数据并选取更优化的参数。

关键词：回归分析、卡方检验、xgboost、非线性整数规划、优化定价体系、R 语言

§1 问题的重述

伴随着经济全球化与互联网平台理念的深入挖掘，企业开始寻求全新的商业模式以冀在市场竞争中赢得有利地位。作为占领市场的先决条件，对市场有全面而细致的认识是企业必须完成的一项评估。传统的市场调研方式局限性较强，企业面临的往往是较高的调查成本与较低精准度的市场评估。相反，在新兴的基于移动互联网的自助式劳务众包平台中，企业可以通过“抢单领赏制”将任务分交给不同地区的用户去完成，这不仅能够大大节省调查成本、缩短调查的周期，更可以有效保证调查数据的真实性。消费者仅需自主在手机软件上领取任务、到达指定地点、按照要求拍照即可领到相应的报酬。因此，如何合理地制定价格以使得消费者能够自主积极地、保质保量地完成任务便成了核心问题。

在“拍照赚钱”软件中，我们发现有不少任务因定价不合理而无人问津，面临商品检查失败的情况。基于已结束项目的任务数据、会员信息数据、新的检查项目任务数据，我们需要完成：

① 寻找已结束项目的任务的定价规律，基于挖掘出的规律分析任务未完成的原因。

② 为已结束项目的任务设计新的定价方案，并论证新方案的优化效果。

③ 考虑到多任务地理位置集中、用户会争相选择的情况，修改定价模型使得其满足任务打包发布的模式，并评估修改后的模型对任务完成度的影响。

④ 针对新的检查项目任务设计新的定价方案，并评价此方案的实施效果。

§2 问题分析

2.1 研究现状综述

众包又称网络社会化生产，是以自由自愿的形式包给非特定的大众网络的做法，具有低成本、联动潜在生产资源、提高生产效率，满足用户个性化需求等优势。众包作为一个新理念，相关文献和研究中还相对缺乏，多以定性研究、客位角度分析为主（Robso&Rew，2010[1]；Alborsa，Ramosb，Herva—sa，2008[2]）一些学者对众包在不同行业的应用进行了探讨，如新闻媒体(吴乐瑁，2007[3]；卫蔚，2010[4])、体育娱乐(石萌萌，2010[5])、饮料(谢园，2010[6])等，但这些研究存在一些不足：一是缺乏有力的经验数据支持。二是对我国的研究现状中大多直接运用西方的众包模式，缺乏针对性。

由此，我们有必要通过数据（来自我国广东省）进行本土化定量研究，通过数学经典模型和机器学习数据挖掘算法，探究众包任务的定价规律及其带来的绩效变化。哪些因素在其中起了关键性作用？如何改进定价机制？如何更好的满足客户个性化需求？在模式和作用

---

[1] N．ROBSON，D．REW．Collective wisdom and decision mak—lag in surgical oncologyEJ]．The Journal of cancer surgery，2010(36)：230-236．

[2] J．ALBORSA．J．C'RAMOSB，J．L HERVASA．New learningnetwork paradigms：communities of objectives，crowdsourcing，wikis andopen source[J]．International Journal of Informa—tion Management。2008(4)：194-202．

[3] 吴乐蘑．“众包”模式推进芙固公民新闻再发展口]．2 际新闻界，2007(8)：40—44．

[4] 卫蔚．新媒体时代国际新闻“解困”之道[J]．采缩谈艺，2010(5)：81—85

[5] 石萌萌．体育娱乐节目——“众包”模式探析[J]．节目栏目．2010(2)：53-55．

[6] 胡泳．众包：业余罔民打败企业超人[J]．中欧商业评论。2009(7)：25—27．

机制的实证探究中，我们考虑了众多因素，如客户的位置信息、信誉度、选配限额、预定时间等结合历史数据分析给出了定量研究结果，进行了合理解释预测，并提出了适用性较好的模型及众包问题的未来发展和研究方向。

2.2 本文的研究思路和步骤

本文研究任务标价方案对任务的完成度的影响，并寻求合理的标价方案使得完成度尽可能地高。对本问题的求解分为 4 个步骤：第一，使用概率模型与深度学习分析已完成任务的信息，寻找任务的定价规律与评估模型。第二，使用深度学习，在满足总支出不增加的情况下调整标价，给出全新的标价方案。第三，使用先排序后分配的方式研究打包发布模式对任务完成度的影响。第四，针对新的项目，分别使用两种模型来评估新标价方案对完成度的优化效果。

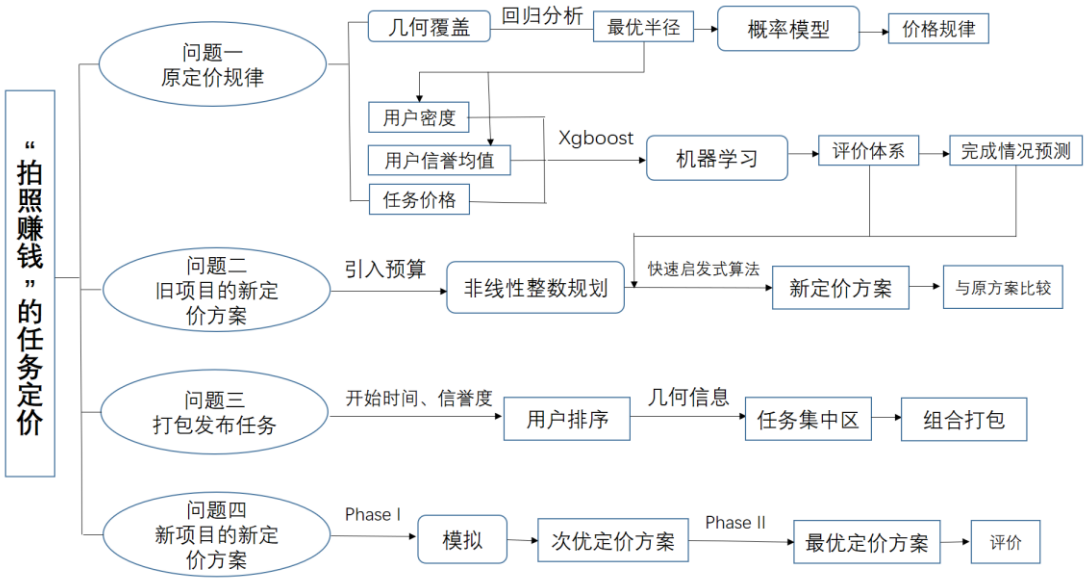

图 1：问题分析流程图

2.3 对具体问题的分析

2.3.1 对问题一的分析

题目要求研究附件一中项目的任务定价规律，并分析任务未完成的原因。

在对数据进行充分分析和讨论后，我们将任务的位置信息转化为任务与用户群体的交互关系，这样做既在理论上合理，数据实际拟合时也得到了很好的结果。具体方式是，首先定位每个任务的位置，利用全体用户数据提取出用户密度[7]、用户信誉均值[8]这两项指标。其次，搭建概率模型，对用户密度、价格的分布进行回归分析，挖掘出了该项目的任务定价规律。

针对本题第二部分分析未完成原因，我们提取了用户密度、用户信誉均值、价格等许多

[7] 用户密度是指以任务为中心一定标准区域内用户的个数。本文模型中该标准区域选为以任务为中心，7km 为半径的圆，后详述。

[8] 即指在该任务为中心的标准区域内用户的信誉均值。

特征进行机器学习，在尝试许多算法后选择了效果最佳的 xgboost，并以此建立评估模型以分析任务未完成的原因。

2.3.2 对问题二的分析

题目要求为原项目设计新定价方案，并对之进行评估。

通过数据模拟[9]和理论分析均可知，同一个任务在标价提升后，其被完成的可能性会提升，因此为使更多任务被完成，必然要提升未被完成任务的标价。一个最简单的想法是将未完成的任务提升到最高价格 85 元。但在实际商业情况中预算是有限的，未必总能粗暴地大幅提价。因此我们引入了预算的概念，表示全部任务的标价之和的一个上界。同时我们引入了以个精细的模型来刻画同时存在价格约束和预算约束时，使任务被完成数尽可能多的优化问题。这是一个非线性整数规划问题，我们分析了它的求解性问题，并对它提出了启发式算法；在用数据实际计算时，被完成的任务数量得到大幅提升，同时整体花费的预算还减少了 1500 元左右。可以说这是一个与原方案相比，非常优越的标价方案。

2.3.3 对问题三的分析

题目要求修改定价模型以适应多任务位置集中这一实际情况，并评估新的定价方式对任务完成度的影响。

我们考虑市场的实际分配情况，给出了较优的打包组合方案。首先，对用户以预订任务开始时间为主要关键字、信誉值为次要关键字进行排序，得到任务分配的优先度次序。其次，根据优先度次序对用户进行遍历，在预订限额的约束下寻找距离最近的任务集中区，得到用户与任务组合的对应情况。进一步，对打包的任务与用户的交互特征提取出来，将其拟合成一个可在 1、2 框架下定价的等价任务，并以此给出其标价。最终，我们用训练好的 xgboost 模型对该定价方案的影响进行分析。

2.3.4 对问题四的分析

题目要求对新项目给出任务定价方案，并对该方案的实施效果进行评估。

基于前三问的基础，本文首先使用问题一中的概率模型给出了一个对任务的初始定价[10]，接着采用问题二中的优化模型的启发式算法进行标价优化。其效果是显著的，并且在有限预算的情况下最大化了任务完成数。我们要指出的是，我们对这个优化问题的处理方法是实现相对简单、计算量较小、有较大普适性的；特别地，在类似于本文研究的这一类有预算约束的最佳定价问题中，该优化模型具有普适性。

§3 模型的假设

1. 我们在模型中仅考虑题中所给出的位置、定价、信誉值等信息，而对于用户偏好、社交影响力、区域经济影响等因素则不纳入考量。
2. 每个用户均为理性人，即在选择任务时寻求个人利益最大化；任务发布者也是理性的，

[9] 用 xgboost 生成价格-完成概率变化图
[10] 以该价格之和作为该项目的任务总预算。

追求以最小代价使最多任务得到完成。

3. 每个任务为公司带来的收益相同的，因此公司为任务标价时应只根据任务的位置、任务与用户群体的（位置）交互关系[11]这两方面因素指定。
4. 不同任务对于用户而言，只有价格、位置的区别，因此对于一个任务是否被完成，只由任务的定价、任务与用户群体的位置交互关系这两方面因素决定。
5. 我们假设一个项目是有预算[12]的，为一个项目设计新的定价方案时，需要控制所有任务的标价之和不超过预算。
6. 标价一般制定在 65-75 之间，只取整数或半整数。存在某些特殊情况可以标价为 80 或者 85 元，并且只允许 65-85 之间的价格。

§4 符号说明

序号	符号	符号说明
1	r	以任务为中心作覆盖的圆半径
2	n_i	任务点 i 的周围（7km 范围内）用户密度
3	d_{tp}	任务位置与用户位置的距离
4	p_i	任务 i 的标价
5	P_{ij}	周围用户密度落在第 j 个区间的任务标第 i 种价格的概率
6	M	用户密度中位数
7	d_f	自由度
8	χ^2	卡方值
9	N	一个项目中的总任务数
10	B	总预算, 满足 $B = \sum_{i=1}^{N} p_i$
11	e_i	任务 i 周围（7km 范围内）用户的信誉均值
12	c_i	任务 i 是否被完成，其中$c_i = 1$表示任务被完成，$c_i = 0$表示任务未被完成
13	pred_xgb_i	xgboost 模型中对于任务 i 被完成概率的预测
14	max_depth	树的最大深度
15	importance_matrix	特征重要性矩阵
16	l_i	用户 i 的任务预订限额

[11] 指任务与用户群体中每个个体的相对位置关系的总和
[12] 以原方案的所有标价之和作为预算。

§5 模型的建立与求解

5.1 问题一的模型建立与求解

5.1.1 对价格规律的探究

众包任务的标价有两种模式：若平台起到整体调节与约束作用，则任务标价将由平台按照一定的规律统一设定；若平台仅起到第三方作用，则任务标价则由发布者自己确定。而无论题中的“拍照赚钱”APP 显然属于哪种，都应该尽力寻求一套合理的定价方案，而在此之前需要首先探究其现有定价规律。根据我们的假设，对任务价格产生影响的只有其位置和与用户群体的位置交互关系。

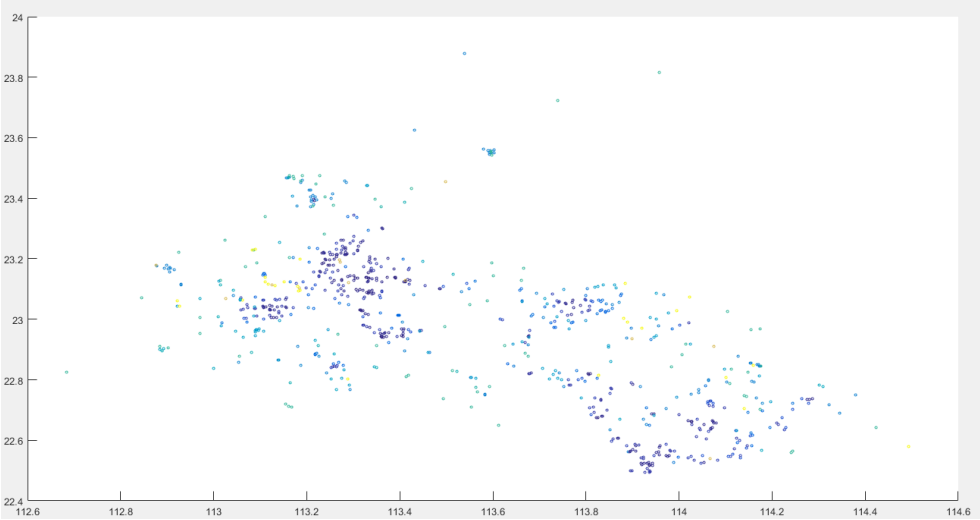

图 2：任务标价与所处位置相关分析图

如图 1 所示，纵轴表示纬度，横轴表示经度，而散点的颜色则表征任务的价格，颜色越深，价格越低。不难发现，紫色、深蓝色的散点（标价较低）分布较为集中，而淡蓝色、绿色、黄色的散点（标价较高）分布较为分散。此外，图像表现出以深色点为中心、浅色点环绕、扩散的模式。

进一步，根据任务的定价分段制图。如图 2 所示，从左到右依次为标价较低（价格小于等于 70 元）的任务整体分布情况、标价较高（价格大于 70 元）的任务局部分布情况（广州市、中山市地区）、标价较高的任务局部分布情况（东莞市、深圳市地区）。观察该图，我们可以进一步印证上述分布结论。

图 3：任务标价与所处位置相关分析图（按价格分段）

基于低价任务集群分布、高价任务扩散分布的情况，我们猜想：任务的价格与位置的相关性源于任务周围用户的分布密度。在用户分布密度比较高的位置，任务的标价相对较低；而在用户分布密度比较低的位置，任务的标价相对较高。为了验证这一猜想，我们将任务数据与用户数据均导入同一张地图。

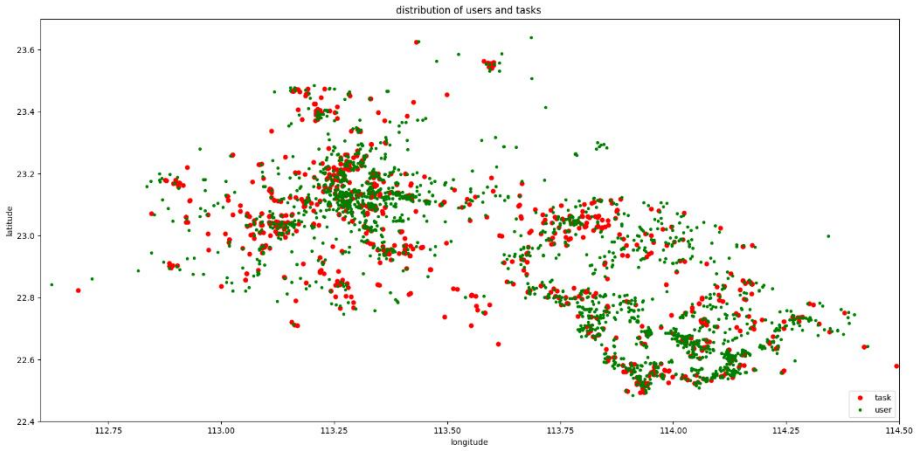

图 4：用户分布与任务分布散点图

由图 3 可知，用户的分布（绿色散点）与任务的分布（红色散点）存在较高的统一性：在任务密度较高的区域，用户密度较高；在任务密度较低的区域，用户密度较低。

为了定量刻画密度这一因素，我们考虑任务周围的几何图形覆盖的用户数量（用户密度）：以任务所处位置为中心，半径为r作圆，我们定义与圆心的距离小于半径长度的用户为任务的周边用户，并通过对周边用户进行计数得出密度值 n。设任务所处位置的经纬度为（x_i, y_i），用户所处位置的经纬度为（a_i, b_i），则两点间的距离d_{tp}满足 Haversine 公式：

$$\sin^2\left(\frac{d_{tp}}{2R}\right) = sin^2\left(\frac{x_i - a_i}{2}\right) + \cos(x_i)\cos(a_i)\,sin^2\left(\frac{y_i - b_i}{2}\right)$$

若$d_{tp} < r$，则我们认为位于经纬度（a_i, b_i）的用户属于任务的周边用户。

以 r=4km 为例，我们给出了不同价格下任务数，周边用户密度的均值、最小值、最大值、中位数的数值[13]。不难发现，随着价格的上升，均值与中位数均呈现出整体递减的趋势。考虑到类似 10 这样的用户密度值存在于各个价格区位中，我们认为用户密度与标价之间满足一种回归性质。

[13] 见附件。

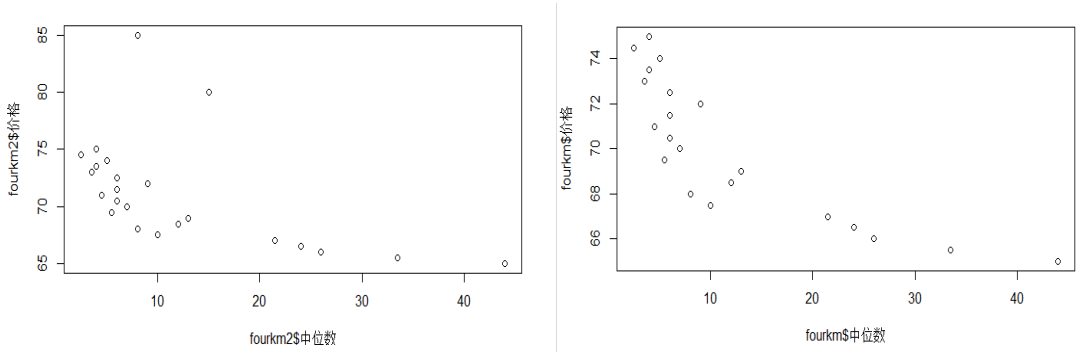

图 5：价格中位数相关性散点图（r=4km）

我们首先生成了 r=1-19km 时的用户密度数据，在实际的观察中，我们发现价格与区间中位数的相关性较为明显，我们分别对 r=1-19km 做了 19 个回归分析，发现 r=7km 时效果最好 Multiple R 值为 0.620185。

但是尽管如此，观察上左图，我们仍不难发现散点图中存在两个偏差较大的点，对应的分别是价格为 80 与价格为 85 的情况。若将这两个点除去，如上右图所示，我们发现价格与中位数呈更好的线性关系。考虑到在价格处于 65~75 的区间中时，价格的落差仅为 0.5 元。而对于 80 与 85 的情况，价格的落差扩大 10 倍，为 5 元。并且其余数据点非常符合线性关系，而这两个点较为异常，同时我们之前在地图上观察时也会发现这两种人物的分布尤其杂乱无章。因此我们推测 65 元~75 元的任务定价较为常规，其价格的落差也比较小；而 80 元与 85 元的任务或许是类似紧急任务的任务，很可能是由于需要用户迅速完成，价格相对设置得较高，因而难以得到严格的线性关系，产生一定偏离。因而我们在研究定价规律时将 80 元和 85 元的数据点删除，仅对 65 元~75 元的数据点进行分析。

将 80-和 85 两点删去后，分别对 r=1-19km 再做回归分析得到下表，可以看到 R-square 值在 r=7 时最高，（具体数据和回归模型详见“1-1 题 1-20km 回归详细.xlsx”），也间接地印证了我们的猜测。事实上，由于任务性质的不同，将 80 元、85 元的任务与 65 元~75 元的任务分开进行处理是符合常理的。而本题并没有给出额外信息，且 80 元、85 元的数据点占整体比重较少，因而我们在下文中仅对主要数据点（65 元~75 元的任务）进行分析。

距离（km）	1	2	3	4	5	6	7	8	9	10
R Square	0.467532	0.581951	0.645309	0.677024	0.739423	0.765421	0.831634	0.806517	0.804772	0.734951
距离（km）	11	12	13	14	15	16	17	18	19	
R Square	0.734951	0.73134	0.725898	0.734323	0.747992	0.7424	0.748012	0.751833	0.771419	

为了再对拟合结果进行优化。考虑到散点图呈略微下凸的弧线状，我们对变量进行取对数处理。而由于价格的落差较小，我们仅对用户密度进行取对数处理，并计用户密度中位数为 M。得到：

$$p = 82.6 - 3.877 * \ln(M) \qquad (1)$$

此时，R-Square= 0.9317。

5.1.2 价格模型的建立

为具体刻画价格规律，我们引入概率定价模型。假设某个任务的用户密度落于区间$[x_j, y_j]$，该区间的中位数为$\mathcal{M}_j$，考虑 logistic 回归模型，该数据点的价格为p_i的概率为$\frac{e^{-\frac{|M_i-\mathcal{M}_j|}{k}}}{\sum e^{-\frac{|M_i-\mathcal{M}_j|}{k}}}$，其中 k 为参数，$M_i = e^{\frac{82.6-p}{3.877}}$为 5.1.1 中提出的公式(1)。K 的选取由理论分析应该不能太大，并遍历 k=5-100 后，发现 k=25 时，较为合适。

为验证此概率模型的可靠性，对它进行卡方检验。首先对于任务按周边用户密度从小到大排序、分组，在满足每组数据点的间隔相差相同的条件下，我们得到十一组区间：[0,19]、[20,39]、[40,59]、[60,79]、[80,99]、[100,119]、[120,139]、[140,159]、[160,179]、[180,199]。经计算，我们可以得出每种价格条件下，各种周围用户密度的概率分布情况（详细见“4 题模拟. xlsx”）。我们再对题中所给附件一的数据进行处理，得到每种价格时，各种周围用户密度的概率分布情况（见附件“4 题模拟. xlsx”）。利用卡方检验的思想，我们将两张表格进行比较，得出在$d_f = 795 - 11^{14} = 784$时，$\chi^2 = 1359.887$。值得一提的是，在 795 个数据点中，有部分数据表现出明显偏大的卡方值。而显然，由于这些数据的存在[15]，模型的总卡方值会偏大。除去卡方值大于 1 的数据点，我们得到$\chi^2 = 758.1404$。此时，P-Value=0.2518。此即表明：给定我们的概率模型为真实分布情况的条件，出现题目中给出的分布情况的概率约为百分之七十五。

以上即说明了我们模型的正确性。我们认为：在分布情况随机的前提下，题中所呈现的情况是可以通过我们的模型假设还原的。换句话说，如果我们的模型是正确的，我们将能以较高的概率观测到题中所呈现的分布情况。这也就同时论证了题中所呈现分布情况的合理性。

综上，

$$\mathrm{P}_{ij} = \frac{\mathrm{e}^{-\frac{|M_i-\mathcal{M}_j|}{25}}}{\sum e^{-\frac{|M_i-\mathcal{M}_j|}{25}}} \qquad (2)$$

5.1.3 评估模型的建立

Xgboost 是梯度提升算法的机器学习函数库，具有优良的学习效果以及高效的训练速度，近年来在各大比赛中作为性能优越的算法受到广泛关注。Xgboost 在 gradient boosting 框架下实现了 tree boosting 算法，代表之一为梯度提升决策树（GBDT）。尤其适合处理分类和回归问题。相比传统算法 GBDT，Xgboost 进行了多项优化，如：对代价函数进行二阶泰勒展开；加正则项防止过拟合等。

在本文讨论的问题中，对于任务完成情况的预测，属于分类问题，非常适合选用机器学习算法进行处理。所以，我们选用 R 语言实现 Xgboost，通过选取比较特征训练出较好的模型。以此进行对任务完成情况的 0, 1 预测。

---

[14] 区间个数为 10。

[15] 大部分数据的卡方值均小于 0.05，而有部分数据的卡方值大于 1.

i. 完成情况模型训练过程

① 特征工程

通过前面阶段的分析，我们计算了任务在方圆 1、2、……20 千米内的用户密度（更现实的反映位置特征），以及任务在方圆 1、2、……20 千米内的信誉均值，并和价格作为 41 个因素（详见 task_info. csv）去训练 xgboost 模型，最终通过共线性分析、重要性矩阵的分析，选取出了 7km 内用户密度和 7km 内信用均值和价格这三个主要因素。通过数据，我们猜测任务完成的情况与任务价格，任务在方圆 7 千米内的用户人数，任务在方圆 7 千米内的用户信誉平均值有关。由此我们选取以上三个特征，进行模型训练。期间，通过对附件一数据的训练集和测试集的随机划分，分类准确率有了提升。

② 调参过程

我们对模型训练进行了多次尝试，一些方式获得了较好的结果，例如加大 max_depth 每棵决策树的最大深度为 20，该参数越大时，越容易过拟合，但由于训练数据较少，该步骤对结果有所增益。

③ 训练结果

Xgboost 输出的结果中包含对每个任务为完成即 1 的预测，结果为 0-1 之间的数字，我们规定大于 0. 5 的预测为完成；并且模型显示了输入特征对最终结果的贡献程度，下图为某次训练过程的 importance_matrix，从中我们可以看到，对完成情况影响顺序从大到小为：7 千米内的用户人数，7 千米内的用户信誉平均值，任务价格。我们的模型分类准确率最终稳定在 75%-83%左右。auc 值在 82-86 左右。

```
> importance_matrix
          Feature      Gain     Cover Frequency
1: `7km_average` 0.4893059 0.4850735 0.4764398
2:  `7km_renshu` 0.3957871 0.3638549 0.3708551
3:         price 0.1149070 0.1510716 0.1527051
```

图 6：特征重要性矩阵

ii. 结果分析——任务未完成原因

由于任务是否完成可以被 xgboost 比较精确地预测，那么任务不完成的原因也应该出在 xgboost 中比较重要的特征之中。调用 R 中的 xgboost 库我们可以得到重要性矩阵。我们在最初训练时加入了价格、1-20km 内的用户密度和 1-20km 内的信誉均值这 41 个变量，经过筛选我们发现最重要的特征与 5. 1. 2. 中线性回归的重要特征一致，最重要的 2 个指标分别是 7km 的信誉均值和 7km 时的用户密度，故而我们在之后的模型中均使用这两个特征和价格作为特征预测完成情况。实际模拟结果显示，用户信誉平均值对任务完成情况的影响程度最大，这比较符合现实情况，任务经常不完成的用户通常信誉度较低，导致了当一个任务周围大多是信誉度较低的会员时，它不能被完成的可能性更大；此外当一个任务周围会员数较少时，任务很难被领取，也会导致任务难以被完成；最后，当一个任务的价格过低时，由于激励不高，会员可能不愿意领取和完成任务。但价格的影响力比我们想象中低，猜测可能是价格的波动较小的原因。我们通过对价格乘方，或对人数、信誉度取对数等方式重新训练模型，得到了类似的结果。说明任务的完成与否主要与周围会员的人数、信誉度和任务价格有

关，尽管价格的影响力没有前两者大，但并不影响我们通过改变价格来使任务被完成的可能性加大，由此我们提出了问题二的方案。

5.2 问题二的模型建立与求解

5.2.1 优化问题的建立

在问题 1 求解的第二部分中我们引入了基于集成随机树模型的 xgboost 算法，对于某个给定标价p_i、周围用户密度n_i和周围用户信誉均值e_i的任务 i，可以产生其pred_xgb_i值，并由pred_xgb_i是否大于 0.5 来判断它是否会被完成，也即得到一个 xgboost 模型（函数），$c_i = c_i(p_i, n_i, e_i)$。

因此，我们可以用该 xgboost 模型对于给定的任务定价方案进行优化，得到一个新的定价方案。我们指出，对于一个项目“设计新的定价方案”是指：控制该项目总体预算不变或减少，并只对每个任务的标价重新设计。

作为一个定价方案的优化问题，我们的目标是在价格和预算约束下，尽可能增加任务的完成数$\sum_{i=1}^{N} c_i(p_i, n_i, e_i)$。将其写为标准的优化问题即是：

$$\text{Max} \sum_{i=1}^{N} c_i(p_i, n_i, e_i)$$

$$s.t. \begin{cases} \sum_{i=1}^{N} p_i \le B \\ 65 \le p_i \le 85, 1 \le i \le N \\ p_i \text{为整数或半整数} \end{cases} \qquad (3)$$

另一方面，由于其目标函数为整数，因此可能存在多种价格方案都能达到其目标函数的最大值，因此在达到相同的任务完成数时，我们还应该尽可能减少预算。

5.2.2 优化问题难度的分析

上述问题由于以下 3 点原因而非常困难：

①由于其目标函数不是解析形式，每一次计算都要重新调用 xgboost 模型，因此梯度下降、牛顿法、内点法等经典显式函数的解法无法使用；而且还不能求得其在某点的一阶信息，所以次梯度法、局部化法和分割算子等办法也不可使用。

②由于我们还不能得知目标函数的凹性，即使对于整数约束进行松弛后依然不是凸问题，求解非常困难。

③其约束条件中第三组条件为整数约束，使问题变为整数规划问题。由整数规划的理论可以知，为精确求解上述问题，其计算复杂度为 NP 难问题，尽管我们问题的规模不大，在有限时间内也绝不可解。

5.2.3 启发式求解算法的提出

面对以上困难，我们首先分析到：由于预算 B 有限，对于部分被完成$(c_i = 1)$的任务，可以适当降低其标价p_i，节省部分预算的同时使它仍然保持完成状态$(c_i = 1)$；这样，对于未完成$(c_i = 0)$的任务 i，就可以有足够预算以提升其标价p_i，使其有可能被完成$(c_i = 1)$。具体操作中，对于有的未完成任务$(c_i = 0)$而言，由于其本身位置等因素，使得即使将其标价提升至上限 85 元时，其pred_xgb_i值依然很小，也即极不可能被完成，我们就将其价格调

低至下限 65 元，以节约预算、或给其他更需要提价的项目进行提价。

依照这个想法，我们提出以下启发式求解算法解决上文提出的优化问题，其效果很好：

①对 835 组数据分别预测其pred_xgb值，并按pred_xgb降序重排。

②对于其中$pred_xgb_i$值大于等于 0.7 的任务 i，每次使其价格p_i减少 0.5 元后重新计算$pred_xgb_i$值，直到以下三个条件之一满足为止：

I. 再降 0.5 则会使其pred_xgb值低于 0.7；

II. 价格已为 65 元；

III. 价格已经降了 5 元[16]。

③并将所有节约出的预算存入 total 对象内。

④让所有pred_xgb值低于 0.5 的任务标价变为 65，将节约的预算也加入 total 中

⑤依序[17]考虑每一个pred_xgb值低于 0.5 任务，每次使其任务标价p_i增加 0.5 元后重新计$pred_xgb_i$值，直到以下 2 个条件之一满足：

I. 其pred_xgb值超过 0.5，此时标记其$c_i = 1$；

II. 其价格p_i达到 85 元。

⑥对于$p_i = 85$但$pred_xgb_i < 0.5$的任务，将其价格重新降至 65 元，节约预算以供更可能被完成的任务提价。

我们的 R 代码附在了支撑材料中，名称为 xgboost_for_problem_2.r。

5.2.4 启发式算法的优化结果：

运行以上启发式算法后，我们得到结果：在我们的 xgboost 模型的判断下，原先有 529 个任务被完成，采用以上定价方案后有 699 个任务被完成，并且总体节约了预算 1506 元（预算的节约是由于，对于优化后的任一未被完成的任务，将其标价提价到 85 元时仍然不能被完成，故而将其标价标为 65 元，以节约整体预算。）

我们的具体标价方案请见 Data1.csv。

5.3 问题三的模型建立与求解

5.3.1 打包制的可操作性分析

在众平台中上用户可以根据各自的权限（即开始预订时间和预订限额）在特定的时间完成特定数额的任务预订。但任务预订的成功与否并不仅仅取决于用户是否完成了预订，还与平台的分配模式相关。换句话说，如果有多位用户在同一时间预订了同一任务，那么该项任务会分配给谁则取决于该平台设立的匹配机制。

我们假设用户均为理性人，行动均符合个人利益的最大化。在此假设的基础上，用户应该在自己可以开始预订的时刻预订等同于自己预订限额数目的任务，而预订任务时的先后顺

[16] 这里条件 I 中不取 0.5 而取 0.7，是出于鲁棒性(robustness)的考虑，为了使原先被完成的任务，在降低标价后仍然有很高的可能性被完成；条件 III 是出于类似的考虑，因为一共只有 20 元的标价浮动区间，某个被完成任务标价的降低幅度不应太大。

[17] 之前已将它们按照$pred_xgb_i$值降序排列过，因此$pred_xgb_i$值低于 0.5 的任务中排序靠前的任务是$pred_xgb_i$值相对较大、也即更有可能被完成的任务，我们优先增加它们的标价。

序则与任务与用户的距离相关，距离越近，任务被该用户预订的优先级就越高。

在传统的分配方式中，平台会综合考虑距离与信誉，将任务分配给距离较近的用户，这导致的结果可能就是相邻的 10 个任务由 10 个人来完成，整体的运行效率较低。而全新的分配模式则考虑任务打包发布，即把较近的任务打包分配给一个用户，由一个用户完成这一个区域的项目。考虑到信誉这一特征的影响，打包制可能使部分用户领取不到任务，但它能使整体运行效率提升，并对任务的完成度产生影响。

下面，我们就将通过建模来进行具体论证。

5.3.2 任务打包组合定价模型的构建

我们对用户以开始预订时间为主要关键字、信誉为次要关键字进行排序。考虑到打包分配制度源于任务的集中性，我们这里反向从用户作为出发点进行考虑：如果在用户的周围紧密地分布着一系列任务，那我们认为这些任务就是集中的。且由于我们对用户的被分配优先度从高到低进行了排序，当我们对用户按此顺序进行遍历时，出现在用户附近的任务一定是分配给该用户的[18]。

基于以上思想，我们对用户—任务进行打包匹配处理。首先我们对每位用户的预订限额进行计算。由于任务分配时实际上是根据预订限额所占比例进行配发的，因此用户 i 在项目中的预订限额l_p为$\frac{l_i}{\sum_j l_j} \times n_t$，其中$n_t$表示该项目的任务总数。

此处我们假设用户认为 2km 以内的任务可以定义为“较近”的任务。以用户为中心，半径为 2km 画圆，依次将圆中的任务点按距离从小到大的优先级与用户进行匹配。若圆中的任务数l_t大于用户可预订的任务上限l_p，则我们仅考虑将距离最近的l_p个点与用户形成匹配；若圆中的任务数l_t小于用户可预订的任务上限l_p，则我们仅考虑圆内的l_t个点。当任务 i 与用户 j 形成匹配后，我们将任务 i 从任务库中除去[19]。

任务数	价格	平均用户密度	平均信誉值	任务编号					
5	67	30	71500.32756	A0537	A0618	A0701	A0743	A0753	
5	65.5	69	61311.89184	A0001	A0003	A0008	A0358	A0360	
6	65.5	102	91698.65088	A0193	A0327	A0348	A0372	A0383	A0461
5	65	68	56888.94082	A0170	A0312	A0318	A0323	A0430	
6	65	56	23312.748	A0521	A0638	A0754	A0770	A0807	A0834
2	69.5	8	9796.11545	A0496	A0763				
5	68	34	9898.8669	A0527	A0646	A0648	A0651	A0653	
4	65	59	6219.545225	A0484	A0490	A0582	A0588		
2	66	62	5407.7984	A0085	A0089				
4	66	37	3877.785575	A0043	A0065	A0077	A0121		
4	67	11	275.771425	A0538	A0666	A0668	A0676		
2	70	8	233.3732	A0292	A0299				
4	65	82	62067.10175	A0005	A0007	A0028	A0033		
5	65	61	16459.00392	A0265	A0376	A0402	A0408	A0412	
2	70	14	501.81	A0212	A0221				
3	66	32	15472.276	A0141	A0151	A0155			
3	65.5	75	60352.55133	A0109	A0119	A0123			
2	65.5	114	12742.95625	A0051	A0060				
2	66	118	11883.5562	A0040	A0075				
2	65	58	6258.8289	A0597	A0602				
3	65.5	65	39847.29767	A0366	A0368	A0427			

图 7：打包模型结果部分图

[18] 基于理性人假设，这个任务只要出现在用户附近，用户就会预订，因此不存在任务离用户较近而用户不预订的情况。

[19] 这是为了避免一项任务匹配多位用户的情况。

注意到，当任务打包后，这一些任务的执行情况将或为均完成，或为均未完成。因此，我们需要考虑将打包的点处理成一个等效点进行运算。以 5 个任务打包为例，我们需要计算以 5 个任务为圆心、7km 为半径的圆中的人物人数，并除以 5 算得均值作为该等效点的周围用户分布密度。同理，我们可以得到该等效点周围的用户的平均信誉。

5.3.3 任务打包组合定价模型的计算

将打包后的等效点与未被打包的“孤立点”组合成新的任务列表，代入问题一中训练的评估模型进行任务完成度评估。值得注意的是：对于打包后的等效点，例如 5 个任务打包后的等效点，若其的评估结果是完成，则在计数时应该记作 5 而不是 1。

最终，模型计算结果显示：任务的完成件数为 516 件，低于初始情况下的 529 件。

我们发现，仅有 43 位用户可以享受打包分配，且打包数均表现为 4 左右[20]。这说明任务主要集中在 43 位用户附近，而当这 43 位优先级较高的用户领取完打包后的任务后，剩下的均为分布较散、离用户较远的任务。因此，打包制虽然确保了集中分布的任务的完成度，使得这一区域的运行效率有所提升，但它将分布较散、距离较远的任务留给了信誉相对较低的用户，这一部分的任务完成度便会下降。换句话说，打包制使得任务完成度本就相对较高的集中分布区的完成度有略微提升，却使得任务完成度本较低的分散分布区的完成度有较大下降，因此从整体上表现出完成度下降的情况。而整体的完成度下降较低[21]，则得益于高信誉用户基数较大，而集群分布的任务基数较小。

综上，我们认为打包制并不是一个合理解决预订冲突的方式，反而会对其整体完成度有一定负面影响。对于用户来说，这更像是一个“贫富差距”加大的过程。

5.4 问题四的模型建立与求解

5.4.1 新项目定价问题与问题一、二、三的联系

对于新项目中的任务进行定价时，我们只知道每个任务的任务编号、经度和纬度。我们需要根据用户信息的数据，以及之前的工作，为这个项目中的任务进行定价，并评估任务的完成情况。

注意到问题一中，探究第一个项目定价规律时，也相当于利用经度、纬度和用户信息数据进行定价。但该定价方案远非最优（见 5.2.），故而为新项目的任务定价时，不该采用一样的定价方法。

而注意到问题二的解决的一个关键即在于已知总体预算和价格制定范围。由于当预算充足时可以将所有价格都提升至上限，必然能达到任务完成数的最大值，但这不符合实际商业追求最大利润的情况。因此问题二中的优化模型，也不能直接使用。

问题三研究的是打包组合定价方案，而结果显示组合方案不够稳定，往往会使得任务完成数降低。由于正如 5.3 中的分析，组合定价的目的往往不是直接变现的利益，而更在于如用户忠诚度等软性层面，由于缺乏大量数据和长时间的观察，无法对该层面的价值进行有效

[20] 在该项目中，拥有最大预订限额的用户可以预订 7 个任务。

[21] $\frac{529-516}{516} = 2.52\%$

建模，故而在新项目中我们暂时不考虑任务的打包组合定价。

5.4.2 新项目定价方案的制定

通过 5.4.1 的分析，很容易得到以下定价方案的思路：

首先利用问题一中的定价规律，用原方案的定价方式（概率模型）给出一个次优的定价方案$\{p_{0,i}\}_{i=1,\dots,N}$。

其次，由 5.2.中的分析，可以知道这个次优的定价方案给是在预算内的，因此我们就得到了预算的一个估计[22]：$B=\sum_{i=1}^{N} p_{0,i}$。而这样我们的定价就可以转化在 5.4.1 中优化问题的框架中，而且还得到了一个可行解$\{p_{0,i}\}_{i=1,\dots,N}$。如此一来，就可以应用 5.4.3 中我们提出的算法求解这一问题。

从优化的角度评价，其实问题一中的次优定价模型相当于一个 PhaseI 方法，为我们的复杂优化问题找到了第一个可行解，因此下称 Phase I。之后的优化相应地称为在 PhaseII，是不断地利用前一步的可行解做变换，最终迭代得到最优解。从这个角度来讲，我们的方法也是经典优化算法的一种变式[23]，我们称之为 PhaseI-PhaseII 优化定价体系。

5.4.3 Phase I:概率模型定价

基于题目所给附件三所包含的位置信息与附件二所包含的用户信息，可以求得新任务周围用户密度和周围信誉均值。我们将任务按用户密度从小到大排序，并以 20 为间隔进行分组，按与 5.1.2 中相同的方法计算各个价格、各用户密度分布下的概率p_{ji}，进而得出基于概率模型的定价方案。

A	B	C	D	E	F	G	H	I	J	K	L
价格\区间	中位数	9.5	29.5	49.5	69.5	89.5	109.5	129.5	149.5	169.5	189.5
65	101	0.072416	0.949559	3.404165	8.771295	72.97613	89.81944	108.849	78.40172	87.1553	40.34263
65.5	74	0.213241	2.796148	10.02417	25.82865	62.18616	30.50228	36.96463	26.62487	29.59755	13.70018
66	60	0.373315	4.895139	17.54905	21.14671	35.5213	17.42318	21.11453	15.20837	16.90639	7.825665
66.5	59	0.38855	5.094913	18.26524	20.31754	34.12849	16.74001	20.28662	14.61204	16.24348	7.518816
67	52.5	0.503922	6.607746	23.68874	15.66587	26.31482	12.90741	15.64203	11.26664	12.52456	5.797395
67.5	39	0.864734	11.33894	17.54905	9.129258	15.33492	7.52177	9.115365	6.565613	7.298666	3.378422
68	34.5	1.035275	13.57517	14.6582	7.625397	12.8088	6.28271	7.613793	5.484061	6.096359	2.821895
68.5	40	0.830827	10.89433	18.26524	9.50183	15.96075	7.828739	9.48737	6.83356	7.596531	3.516298
69	39	0.864734	11.33894	17.54905	9.129258	15.33492	7.52177	9.115365	6.565613	7.298666	3.378422
69.5	29.5	1.264487	16.58075	12.00112	6.243147	10.48696	5.143848	6.233646	4.489969	4.991276	2.310372
70	19	1.924501	10.89433	7.885295	4.10204	6.890423	3.379749	4.095798	2.95012	3.279502	1.518023
70.5	19	1.924501	10.89433	7.885295	4.10204	6.890423	3.379749	4.095798	2.95012	3.279502	1.518023
71	30	1.239449	16.25243	12.24355	6.369267	10.69881	5.247761	6.359574	4.580672	5.092107	2.357045
71.5	22	1.706879	12.28332	8.890645	4.625037	7.76893	3.810656	4.617999	3.326251	3.697628	1.711566
72	24	1.575648	13.30637	9.631121	5.010243	8.415981	4.128035	5.002618	3.603285	4.005593	1.854117
72.5	16	2.169869	9.662405	6.993629	3.638183	6.111257	2.997568	3.632647	2.616522	2.908658	1.346365
73	13	2.44652	8.569785	6.202793	3.226779	5.420199	2.658605	3.221868	2.320646	2.579748	1.194119
73.5	8	2.650283	7.016346	5.078417	2.641863	4.437683	2.176681	2.637843	1.899985	2.112119	0.977662
74	8	2.650283	7.016346	5.078417	2.641863	4.437683	2.176681	2.637843	1.899985	2.112119	0.977662
74.5	8	2.650283	7.016346	5.078417	2.641863	4.437683	2.176681	2.637843	1.899985	2.112119	0.977662
75	8	2.650283	7.016346	5.078417	2.641863	4.437683	2.176681	2.637843	1.899985	2.112119	0.977662

图 8：新任务下的概率模型计算结果

上图给出了区间为[0,19]、[20,39]、[40,59]、[60,79]、[80,99]、[100,119]、[120,139]、[140,159]、[160,179]、[180,199]时的任务分布情况。图中的第一纵行表示价格，

[22] 实际上这里的 B 是预算的下界，但是我们用更小的可行域去进行优化，实际上等价于是一种鲁棒优化的做法，可以得到更为鲁棒的结果。

[23] 比如内点法也是这样类型的算法，先用 PhaseI 找到一个在可行域内的点，PhaseII 再进行优化

考虑到定价区间的固定性，我们这里仍采用 65~75 的区段。图中的第二纵行表示价格对应的中位数。图中第一横行的数据则表示各个区间的中位数。而在剩余的数字则表示：基于概率模型的结论下，各个位置的任务数量。为构造方便，我们对每项数值进行高斯取整，最终对相对整数缺失的部分进行随机分布处理，最终得到一个可作为 Phase II 中初始可行解的任务定价方案$\{p_{0,i}\}_{i=1,\dots,N}$。具体的价格数据表见支持文件 problem_4_Phase_I. csv。

5.4.4 Phase II：价格方案优化

我们将 5.4.3.中得到的定价方案的周围用户分布密度、信誉均值与标价用 5.1.中训练得到的 xgboost 模型进行完成情况的预测，得到以下结果：在 2066 个任务中，有 899 个任务可以被完成，剩余的 1167 个任务未完成。（详见 Data_problem_4_Phase_I. csv）因此，基于概率模型的定价体系，该项新任务的任务完成度为$\frac{899}{899+1167}=43.5\%$。

将该定价方案代入 5.4.3 中的优化算法进行优化，得到定价方法优化效果非常明显：有 1513 个任务被完成，只有 553 个项目未完成，完成度提升至 72.3%，并且节省了 1101.5 元预算。（新定价方案详见 Data_problem_4_Phase_II. csv）

§6 误差分析与灵敏性检验

6.1 误差分析

1. 建立概率模型时，各项中位数是基于已完成任务的情况得出的，这与使用概率模型计算得出的理论值存在偏差。
2. 建立评价模型时，深度学习的结果在小范围内存在浮动，造成一定误差。
3. 数据处理中使用了取整等处理方式，造成数据与理论值有所偏差。

6.2 灵敏度分析

1.评价模型的稳定性，由于 xgboost 是一个随机算法，在数据量较小的时候得到的结果会有小幅波动，但是整体结果在交叉验证中可以看出效果很好，相对比较稳定。

2.概率定价模型想法很新，但是不够稳定；但是我们主要的定价是依靠优化定价体系给出的，概率定价的方法只是提供一个 PhaseI 的可行解，而优化方法本身的稳定性依赖于 xgboost 对于优化目标刻画的准确程度，因此随着数据集大小的增长，我们的定价也是非常稳定的。

§7 模型的评价与推广

7.1 优缺点分析

7.1.1 优点分析

- 我们的模型结合了许多不同领域内或前沿(xgboost、非线性整数规划等)或经典(回归分析等)的方法，为这一新型定价问题提供了丰富而有效的定价方法和分析评价方法。

- 使用概率模型而不是简单的线性回归，能较好地解释定价的规律。
- 使用机器学习的方式进行任务完成的评估，且使用了题中所给的原始数据作为训练材料，结果具有较强的说服力。
- 创新性地意识到可以通过添加预算约束这一条件，将定价问题转变为优化问题，除了方法本身的优美、实践效果好等优势外，还提出了关于节省预算开销这一新的考虑方面。

7.1.2 缺点分析

- 模型中的部分参数为人为设定，因而存在一定的主观性。
- 在特征选择的时候只选取了 41 个交互信息变量进行特征工程，不能说做的非常精细。
- xgboost 本身是一个随机算法，所以每次结果会由于数据量的有限而产生小幅波动。
- 原始数据量有限，并且时间有限，不能把结果做的十分准确、参数全部调到最佳。

7.2 模型推广

在本文的探究中，我们采用了概率模型和机器学习模型，可以对价格进行合理的判断，并对由价格的变动带来的任务完成情况变化进行较准确的预测。这解决了在众包问题的运作机制中广泛关注的几个要点：

1）影响因素众多，机器学习模型在处理多种特的分类时具有极大的优越性，方便企业把这些因素数据化，提取特征，得到因素可能的影响因子；

2）预测定价作用机制，概率模型能够对定价给出合理依据，扩大行业受益，提高运营效率；同时，公司、企业可以利用机器学习模型对准备做出的定价调整预测可能产生的市场效果；

3）满足客户个性化需求，我们考虑用户信誉、位置、时间等因素，对任务进行打包处理，最大可能的适应市场和用户需求，有助于使大众参与促进行业发展。

在大数据兴盛的今天，数学模型、机器学习对我们研究众包问题提供了许多便利。未来，我们可以通过对用户及任务历史数据的研究，量化用户个人偏好和任务难度、关注度等有较深探讨和挖掘价值的因素，来更好的修正定价方案。正确掌握众包研究的规律和动态，对于我国开放式创新的深入开展、服务企业创新战略的实施具有重要的理论和现实意义。

§8 模型的改进

8.1 概率模型的改进

在问题一中，我们拟合出了各价格与相应中位数所满足的线性关系。而我们在后续的模型中并没有用到此步拟得的线性关系，而是通过 logistic 回归的方式去搭建概率模型。这在一定程度上基于我们对价格梯度的固定。而事实上，由于线上支付的便携性，诸如 0.4、0.3、0.2 等尾数的价格在定价中也是非常常见。因此，我们可以充分利用我们得到的线性方程$p = 82.6 - 3.877 * \ln(M)$，进行连续的价格设定。在连续的价格设定上进行位数保留处理，我们将会得到梯度更小的价格设定。而基于新的价格设定，我们得到的概率分布将会更加细致、精准。之后，我们再将价格以 0.5 为梯度进行归类（例如 65.2 则归为 65；65,4 则归为 65.5）。

在归类时概率也进行相应的求和。

经过这样处理的概率模型较之我们在论文中给出的将会更加精准，更契合题中所给的数据规律。

8.2 打包模型的改进

在问题三中，我们通过以用户位置为中心，半径为 2 画圆，寻找落在这个几何覆盖中的点作为寻找打包任务的方式。这种寻找方式可以找到的是离用户最近的点，但却不是用户遍历一遍最短的点。

假设用户充分理性，那么用户需求的一定是单位任务所需长度最短。假设用户 i 可以选择l_i个点，我们队几何覆盖中的所有点进行评估，寻求一种单位任务长度最短的模式。单位任务长度最短，即遍历最短路径与任务数的商最小。换句话说，我们需要对从 1 到l_i的选择数进行评估：给定用户位置与被选择的任务点的位置，利用 Dijkstra 算法，我们能得出最短路径。将所有这样的解进行比较，我们便能得出“最理性”的打包情况。

8.3 优化定价模型的改进

由之前的敏感性分析可以看到，优化模型的准确性一方面来自于 xgboost 对于完成情况预测的准确程度，另一方面来自于启发式算法对于原问题真实最优解的逼近程度。前者我们可以通过在之后多收集相关数据，减小由于小样本带来的随机误差；对于后者，现在暂时还没有相应的理论保证，虽然在本文的两个例子中，该算法的效果很不错，但是仍然需要进一步寻求理论上的分析与保障。

§9 参考文献

[1] 孙信昕. 众包环境下的任务分配技术研究[D]. 扬州大学, 2016.

[2] 许春娇. 基于众包模式的城市事件分析系统[D]. 哈尔滨工业大学, 2016.

[3] 雒兴刚, 汪定伟, 唐加福,等. 软件开发项目中任务调度的混沌遗传算法[J]. 小型微型计算机系统, 2006, 27(10):1923-1926.

[4] 刘晓钢. 众包中任务发布者出价行为的影响因素研究[D]. 重庆大学, 2012.

[5] 陈家银. 猪八戒众包平台数据分析与众包模式设计[D]. 大连理工大学, 2016.

[6] 吴昌钱, 洪欣. 一种改进的基于社团发现的贝叶斯众包模型[J]. 湘潭大学自科学报, 2015(4):87-91.

[7] 黄丽媚. 基于众包平台的用户信用度及消息推送算法研究[D]. 广东技术师范学院, 2016.

[8] 徐雅卿, 魏轶华, 胡奇英. 基于 Priceline 的买方/卖方定价收益管理问题[J]. 管理科学学报, 2008, 11(3):63-69.

[9] 宋天舒, 童咏昕, 王立斌,等. 空间众包环境下的 3 类对象在线任务分配[J]. 软件学报, 2017, 28(03):611-630.

[10] 毛宁. 求解离散优化问题的人工蜂群算法研究[D]. 大连海事大学, 2016.
[11] 宋天舒, 童咏昕, 王立斌,等. 空间众包环境下的 3 类对象在线任务分配[J]. 软件学报, 2017, 28(03):611-630.
[12] 张亭亭. 基于成对关联属性空间的众包任务优化配置及其关键属性选择[D]. 江苏科技大学, 2014.

§10 附录

10.1 代码

10.1.1 问题一的代码 1

```
    # -*- coding: utf-8 -*-

import os

#写入文件
f=open("output_task_user.txt","w")

task=[]#纬度，经度，价格
user=[]#纬度，经度
for line in open("task.txt","r").readlines():
    #保存一个空格连接
    result1=' '.join(line.split())
    #获取每行值
    s1=[float(x) for x in result1.strip().split(' ')]
    #数据存储到 dtask
    task.append(s1)

for line in open("user.txt","r").readlines():
    #保存一个空格连接
    result2=' '.join(line.split())
    #获取每行值
    s2=[float(x) for x in result2.strip().split(' ')]
    #数据存储到 user
    user.append(s2)

from math import radians, cos, sin, asin, sqrt,fabs
def hav(theta):
    s=sin(theta/2)
    return s*s

def dis(lon1,lat1,lon2,lat2):#经度 1，纬度 1

    #十进制转化为弧度
    lon1,lat1,lon2,lat2 = map(radians,[lon1,lat1,lon2,lat2])
    #haversine 公式
    dlon=fabs(lon2-lon1)
    dlat=fabs(lat2-lat1)
    h=hav(dlat)+cos(lat1)*cos(lat2)*hav(dlon)
    r=6371  #地球半径
    distance=2*r*asin(sqrt(h))     #千米
```

```
    return distance

#遍历 task，计算用户密集度

k_test=1
while k_test<20:
    num = 0
    data = [[65, 0, []], [65.5, 0, []], [66, 0, []], [66.5, 0, []], [67, 0,
[]], [67.5, 0, []], [68, 0, []], [68.5, 0, []], [69, 0, []], [69.5, 0, []],
[70, 0, []], \
            [70.5, 0, []], [71, 0, []], [71.5, 0, []], [72, 0, []], [72.5, 0,
[]], [73, 0, []], [73.5, 0, []],[74, 0, []], [74.5, 0, []], [75, 0, []], [80,
0, []], [85, 0, []]]

    for x in task:
        k=int(x[2]*2)
        if x[2]==80:
            data[21][1]+=1
        elif x[2]==85:
            data[22][1]+=1
        else:
            data[k-130][1]+=1

        n=0
        for y in user:
            distance=dis(x[1],x[0],y[1],y[0])
            if (distance< k_test):
                n+=1
            num+=1

        k=int(x[2]*2)
        if x[2]==80:
            data[21][2].append(n)
        elif x[2]==85:
            data[22][2].append(n)
        else:
            data[k-130][2].append(n)


    #计算次数
    print(num)
    #对不同价格的任务用户数排序
    print("价格 任务数 对应任务在%d km 内的用户数"%k_test,file=f)
    for x in data:
```

```
        x[2]=sorted(x[2])
        print(x[0], end=' ', file=f)
        print(x[1], end=' ', file=f)
        for y in x[2]:
            print(y, end=' ',file=f)
        print('',file=f)

    k_test+=1
```

10.1.2 问题一的代码 2

```
    # -*- coding: utf-8 -*-

import os

#写入文件
f=open("output_price_distribution.txt","w")

price=[]

#读入文件
for line in open("price.txt","r").readlines():
    #保存一个空格连接
    result1=' '.join(line.split())
    #获取每行值
    s=[float(x) for x in result1.strip().split(' ')]
    #数据存储到 price
    price.append(s)

distribution=[]
i=0
while i< 21:
    s=[price[i][0],0,0,0,0,0,0,0,0,0,0]#价格，人数……
    distribution.append(s)
    i+=1


j=0#价格 65-75
for x in price:
    i=0#从第三列数据开始遍历
    for y in x:
        if i>1:
            if (y>=0 and y<=19):
                distribution[j][1]+=1
            elif(y>=20 and y<=39):
                distribution[j][2] += 1
```

```
            elif(y>=40 and y<=59):
                distribution[j][3] += 1
            elif(y>=60 and y<=79):
                distribution[j][4] += 1
            elif(y>=80 and y<=99):
                distribution[j][5] += 1
            elif(y>=100 and y<=119):
                distribution[j][6] += 1
            elif(y>=120 and y<=139):
                distribution[j][7] += 1
            elif(y>=140 and y<=159):
                distribution[j][8] += 1
            elif(y>=160 and y<=179):
                distribution[j][9] += 1
            elif(y>=180 and y<=199):
                distribution[j][10] += 1
        i+=1
    j+=1


print("价格\区间",file=f)

for x in distribution:
    for y in x:
        print(y,end=' ',file=f)
    print('', file=f)

f.close()
```

10.1.3 问题一的代码 3

```
    # -*- coding: utf-8 -*-

import os

#写入文件
f=open("output_task_credit.txt","w")

task=[]#纬度，经度，价格，完成情况
user_cre=[]#纬度，经度，信誉

#读入文件
for line in open("task_com.txt","r").readlines():
    #保存一个空格连接
    result1=' '.join(line.split())
```

```
    #获取每行值
    s1=[float(x) for x in result1.strip().split(' ')]
    #数据存储到 dtask
    task.append(s1)

for line in open("user_credit.txt","r").readlines():
    #保存一个空格连接
    result2=' '.join(line.split())
    #获取每行值
    s2=[float(x) for x in result2.strip().split(' ')]
    #数据存储到 user
    user_cre.append(s2)

from math import radians, cos, sin, asin, sqrt,fabs
def hav(theta):
    s=sin(theta/2)
    return s*s
def dis(lon1,lat1,lon2,lat2):#经度 1，纬度 1

    #十进制转化为弧度
    lon1,lat1,lon2,lat2 = map(radians,[lon1,lat1,lon2,lat2])
    #haversine 公式
    dlon=fabs(lon2-lon1)
    dlat=fabs(lat2-lat1)
    h=hav(dlat)+cos(lat1)*cos(lat2)*hav(dlon)
    r=6371  #地球半径
    distance=2*r*asin(sqrt(h))     #千米
    return distance

task_com=[]#标号，价格，完成情况，7km 内信誉均值

#遍历 task，找出 7KM 内用户信誉值均值
i=0
for x in task:
    total_cre=0
    n=0
    a = []#一条任务信息
    a.append(i)#任务标号
    a.append(x[2])#价格
    a.append(x[3])#完成情况

    for y in user_cre:
        distance = dis(x[1], x[0], y[1], y[0])
        if (distance < 7):
```

```
            total_cre+=y[2]
            n+=1
    if n==0:
        average=0
    else:
        average=total_cre/n
    a.append(average)#信誉均值
    task_com.append(a)
    i+=1

print("任务序号 价格 完成情况 7km 内用户信誉均值", file=f)
i=0
for x in task_com:
        print(x[0], end=' ', file=f)
        print(x[1], end=' ', file=f)
        print(x[2], end=' ', file=f)
        print(x[3],end=' ',file=f)
        print('',file=f)

f.close()
```

10.1.4 问题三的代码

```
    # -*- coding: utf-8 -*-

import os

#写入文件
f1=open("output_bunding_user.txt","w")
f2=open("output_bunding.txt","w")

user=[]#编号，纬度，经度，预定限额，开始时间，信誉值，实际限额
task=[]#编号，纬度，经度，价格
result=[]#输出结果 任务数 价格 单位任务路径

#读入文件
for line in open("task_order.txt","r").readlines():
    #保存一个空格连接
    result1=' '.join(line.split())
    #获取每行值
    s1=[x for x in result1.strip().split(' ')]
    #数据存储到 task
    task.append(s1)

for line in open("user_order.txt","r").readlines():
```

```
    #保存一个空格连接
    result2=' '.join(line.split())
    #获取每行值
    s2=[x for x in result2.strip().split(' ')]
    #数据存储到 user
    user.append(s2)

from math import radians, cos, sin, asin, sqrt,fabs
def hav(theta):
    s=sin(theta/2)
    return s*s

def dis(lon1,lat1,lon2,lat2):#经度 1，纬度 1

    #十进制转化为弧度
    lon1,lat1,lon2,lat2 = map(radians,[lon1,lat1,lon2,lat2])
    #haversine 公式
    dlon=fabs(lon2-lon1)
    dlat=fabs(lat2-lat1)
    h=hav(dlat)+cos(lat1)*cos(lat2)*hav(dlon)
    r=6371  #地球半径
    distance=2*r*asin(sqrt(h))     #千米
    return distance

#计算 7km 内用户人数和信誉均值
tnum=[]
dist=[]
cred=[]
for x in task:
    ndist=0
    ncred=0
    tnum.append(x[0])
    for y in user:
        distance = dis(float(x[2]), float(x[1]), float(y[2]), float(y[1]))
        if(distance<7):
            ndist+=1
            ncred+=float(y[5])
    dist.append(ndist)
    cred.append(ncred)

dic_d=dict(zip(tnum,dist))
dic_c=dict(zip(tnum,cred))
```

```
total_user=[]
i=0
for x in user:
    s=[]
    tasknum=int(x[6])
    for y in task:
        point = []  # 2km 内任务编号，距离
        if(tasknum==0 or tasknum==1):
            break
        distance = dis(float(x[2]), float(x[1]), float(y[2]), float(y[1]))
        if(distance<2):
            point.append(y[0])
            point.append(distance)
            point.append(y[3])
            s.append(point)#用户

    sorted(s, key=lambda s: s[1])#按距离排序
    if(len(s) > tasknum):#2km 内任务比限额大,取限额数内最近的一些未打包的任务打
包
        s=s[:tasknum:1]
    elif(len(s)==1):#一个任务不能打包
        s=[]

    s.insert(0,user[i][0])#用户编号,打包[任务编号，距离,价格]
    total_user.append(s)

    #删除已打包的任务
    task_bunding = []#打包的编号
    total_price=0
    total_cred=0
    total_num=0
    for a in task:
        for b in s:
            if(a[0]==b[0]):
                task.remove(a)
                total_price+=int(float(a[3]))
                task_bunding.append(a[0])#加入打包任务组
    if(task_bunding!=[]):
        t=0
        while(t<len(task_bunding)):
            total_cred+=dic_c[task_bunding[t]]
            total_num+=dic_d[task_bunding[t]]
            t+=1
        aver_num=total_num/len(task_bunding)
```

```
        aver_cre = total_cred / len(task_bunding)

temp=[len(task_bunding),task_bunding,round(total_price/len(task_bunding),1),ave
r_num, aver_cre]
        result.append(temp)
    i+=1



print("任务数 价格 平均用户密度 平均信誉值 任务编号  ",file=f2)
for x in result:
    print(x[0],end=' ',file=f2)
    print(((x[2]+0.25)//0.5)*0.5,end=' ',file=f2)
    print(int(x[3]), end=' ', file=f2)
    print(x[4], end=' ', file=f2)
    for y in x[1]:
        print(y,end=' ',file=f2)
    print('',file=f2)

for x in task:
    print("1",end=' ',file=f2)
    print(((float(x[3])+0.25)//0.5)*0.5, end=' ', file=f2)
    print(int(dic_d[x[0]]),end=' ',file=f2)
    print(dic_c[x[0]],end=' ',file=f2)
    print(x[0], file=f2)



print("用户打包信息：用户编号，[打包任务编号，距离，价格]",file=f1)
for x in total_user:
    for y in x:
        print(y,end=' ',file=f1)
    print('',file=f1)
```

10.1.5 问题四的代码 1

```
    # -*- coding: utf-8 -*-

import os

#写入文件
f=open("output_new_task_user.txt","w")

task=[]#纬度，经度
user=[]#纬度，经度
for line in open("task_new.txt","r").readlines():
```

```
    #保存一个空格连接
    result1=' '.join(line.split())
    #获取每行值
    s1=[float(x) for x in result1.strip().split(' ')]
    #数据存储到 dtask
    task.append(s1)

for line in open("user.txt","r").readlines():
    #保存一个空格连接
    result2=' '.join(line.split())
    #获取每行值
    s2=[float(x) for x in result2.strip().split(' ')]
    #数据存储到 user
    user.append(s2)

from math import radians, cos, sin, asin, sqrt,fabs
def hav(theta):
    s=sin(theta/2)
    return s*s

def dis(lon1,lat1,lon2,lat2):#经度 1，纬度 1

    #十进制转化为弧度
    lon1,lat1,lon2,lat2 = map(radians,[lon1,lat1,lon2,lat2])
    #haversine 公式
    dlon=fabs(lon2-lon1)
    dlat=fabs(lat2-lat1)
    h=hav(dlat)+cos(lat1)*cos(lat2)*hav(dlon)
    r=6371  #地球半径
    distance=2*r*asin(sqrt(h))     #千米
    return distance

#遍历 task，计算用户密集度

k_test=7

for x in task:
    n=0
    for y in user:
        distance=dis(x[1],x[0],y[1],y[0])
        if (distance< k_test):
            n+=1
    x.append(n)
```

```
#对不同价格的任务用户数排序
print("纬度 经度 对应任务在%d km 内的用户数"%k_test,file=f)
for x in task:
    print(x[0], end=' ', file=f)
    print(x[1], end=' ', file=f)
    print(x[2], end=' ',file=f)
    print('',file=f)

f.close()
```

10.1.6 问题四的代码 2

```
    # -*- coding: utf-8 -*-

import os
import random

#写入文件
f=open("output_intimate.txt","w")

prop=[]#十个区间概率值

for line in open("intimate.txt","r").readlines():
    #保存一个空格连接
    result1=' '.join(line.split())
    #获取每行值
    s2=[int(float(x)) for x in result1.strip().split(' ')]
    #数据存储到 dtask
    prop.append(s2)


total=[30,194,233,175,371,236,286,206,229,106]
n=0
for x in prop:
    s = []
    m=0
    for y in x:
        num=y
        i=0
        while(i<num):
            s.append(int(m*0.5+65))
            i+=1
        m+=1

    if(len(s)<total[n]):
```

```
            k=total[n]-len(s)
            b=0
            while(b<k):
                s.append(random.randint(0,20)*0.5+65)
                b=b+1

        n+=1

        random.shuffle(s)
        for x in s:
            print(x,file=f)


f.close()
```

10.1.7 画图代码

```
    """
可视化绘图

"""
import os
import numpy as np
import matplotlib.pyplot as plt

task=[]#纬度，经度，价格
user=[]#纬度，经度
for line in open("task.txt","r").readlines():
    #保存一个空格连接
    result1=' '.join(line.split())
    #获取每行值
    s1=[float(x) for x in result1.strip().split(' ')]
    #数据存储到 dtask
    task.append(s1)

for line in open("user.txt","r").readlines():
    #保存一个空格连接
    result2=' '.join(line.split())
    #获取每行值
    s2=[float(x) for x in result2.strip().split(' ')]
    #数据存储到 user
    user.append(s2)

from math import radians, cos, sin, asin, sqrt,fabs
def hav(theta):
```

```
    s=sin(theta/2)
    return s*s

#获取第一列和第二列数据
tx=[n[0] for n in task]
ty=[n[1] for n in task]

ux=[n[0] for n in user]
uy=[n[1] for n in user]

#绘制散点图


plt.xlim(xmax=114.5,xmin=112.6)
plt.ylim(ymax=23.7,ymin=22.4)
#四个颜色：红 绿 蓝 黄
plot1, = plt.plot(ty,tx,'or',marker="o",markersize=5)
plot2, = plt.plot(uy,ux,'og',marker="o",markersize=3)


#标题
plt.title("distribution of users and tasks ")

#x 和 y 轴坐标
plt.xlabel("longitude")
plt.ylabel("latitude")

#图例


plt.legend((plot1,plot2),('task','user'),fontsize=10)
plt.show()
```

10.1.8 R 代码

```
library(xgboost)

library(ROCR)

# Data<-subset(task_info,select = -
c(task,alt,longt,`16km_renshu`,`7km_renshu`,`8km_renshu`,`15km_renshu`,

#
`5km_renshu`,`9km_renshu`,`17km_renshu`,`4km_renshu`,`1km_renshu`,
```

```
#
`10km_renshu`,`11km_renshu`,`6km_renshu`,`20km_renshu`,`3km_renshu`,

#
`2km_average`,`8km_renshu`,`3km_renshu`,`7km_renshu`,`10km_renshu`,

#
`11km_renshu`,`4km_renshu`,`9km_renshu`,`6km_renshu`,`1km_renshu`,

#
`5km_renshu`,`17km_renshu`,`20km_renshu`,`12km_renshu`,`12km_average`,

#
`3km_average`,`15km_average`,`13km_average`,`1km_average`,`11km_average`,

#
`4km_average`,`14km_renshu`,`19km_renshu`,`2km_renshu`))




Data<-subset(task_info,select =
c(task,price,`7km_renshu`,`7km_average`,outcomes))



# for (i in 1:835){

#   Data$`7km_renshu`[i]<-log(1+Data$`7km_renshu`[i])

#   Data$`7km_average`[i]<-log(1+Data$`7km_average`[i])

#   Data$price[i]<-(Data$price[i]-65)^5

# }

##5-fold cv for xgboost

index = sample(1:835,835)

cut = c(1,168,335,502,669,835)

accu.xgb<-rep(0,5)

auc.xgb<-rep(0,5)

for(i in 1:5){
```

```
Data.train<-Data[-index[cut[i]:(cut[i+1]-1)],]

Data.test<-Data[index[cut[i]:(cut[i+1]-1)],]


train_x <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data.train)

head(train_x[,1])

xgb <- xgboost(data=data.matrix(train_x[,-1]),

                label=Data.train$outcomes,

                eta=0.2,

                max_depth=20,

                colsample_bytree=0.8,

                nrounds=110,

                seed=1,

                eval_metrix="error",

                objective="binary:logistic",

                nthread=7)


Test <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data.test)

pred_xgb <- predict(xgb,Test[,-1])

prediction_xgb <- prediction(pred_xgb,Data.test$outcomes)

perf_xgb <- performance(prediction_xgb,"tpr","fpr")

plot(perf_xgb,colorize=T)

auc_xgb <- performance(prediction_xgb, measure = "auc")

auc_xgb <- auc_xgb@y.values[[1]]

auc.xgb[i]=auc_xgb

##
```

```
  names <- colnames(train_x[,-1])
  importance_matrix <- xgb.importance(names,model=xgb)
  ##
  threshold <- 0.5
  result = vector(length=nrow(Data.test))
  result[pred_xgb < threshold] <- 0
  result[pred_xgb >= threshold] <- 1
  ##
  TP = sum(result==1&Data.test$outcomes==1)
  FN = sum(result==0&Data.test$outcomes==1)
  FP = sum(result==1&Data.test$outcomes==0)
  TN = sum(result==0&Data.test$outcomes==0)
  ##
  P=TP/(TP+FP)
  R=TP/(TP+FN)
  ##
  accu <- (TP+TN)/nrow(Data.test)
  accu.xgb[i] = accu
}
mean(accu.xgb)
mean(auc.xgb)

##train model with full set
train_x <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data)
head(train_x[,1])
```

```
xgb <- xgboost(data=data.matrix(train_x[,-1]),
                label=Data$outcomes,
                eta=0.2,
                #min_child_weight=0.5,
                max_depth=10,
                colsample_bytree=0.8,
                nrounds=100,
                seed=1,
                eval_metrix="error",
                objective="binary:logistic",
                nthread=7)
Test <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data)
Data$pred_xgb <- predict(xgb,Test[,-1])



############################solving problem 2
##sort data
Data1<- Data[order(-Data$pred_xgb),]


##adjust price
# lm.sol<-lm(price~log(1+`7km_renshu`)+log(1+`7km_average`),Data)
# lm.sol<-lm(price~.,Data)
# summary(lm.sol)
```

```
##decrease price of pred_xgb>0.5
i=1
total=0
while(i<836){
  #record price
  price<-Data1[i,]$price
  ##decline 0.5 price
  while(Data1[i,]$pred_xgb>0.7){
    Data1[i,]$price<-Data1[i,]$price-0.5
    #test the try of adjust price
    Test <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data1)
    pred_xgb <- predict(xgb,Test[,-1])
    if(pred_xgb[i]<0.7|Data1[i,]$price<65|price-Data1[i,]$price>5) {
      Data1$price[i]<-Data1$price[i]+0.5
      total<-total+price-Data1$price[i]
      break
    }
    Data1[i,]$pred_xgb<-pred_xgb[i]
  }
  i<-i+1
}

##add price for pred_xgb<0.5

#first decrease to 65
```

```
for (i in 1:835){
  if(Data1[i,]$pred_xgb<0.5){
    total<-total+Data1[i,]$price-65
    Data1[i,]$price<-65
  }
}



#then incease as for the largest pred_xgb's price
i=1
while(i<836){
  ##add 0.5 price each turn
  while(Data1[i,]$pred_xgb<0.5){
    Data1[i,]$price<-Data1[i,]$price+0.5
    #test the try of adjust price
    Test <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data1)
    pred_xgb <- predict(xgb,Test[,-1])
    Data1[i,]$pred_xgb<-min(pred_xgb[i]*1)
    if(Data1[i,]$pred_xgb>=0.5|Data1[i,]$price>=85) {
      total<-total+65-Data1$price[i]
      break
    }
  }
  if (Data1[i,]$pred_xgb<0.5){
    total<-total+Data1[i,]$price-65
```

```
    Data1[i,]$price<-65


  }
  if (total<0) {
    total<-total-65+Data1[i,]$price
    Data1[i,]$price<-65
    break
  }


  i<-i+1
}


for (i in 1:835){
  if(Data1[i,]$pred_xgb>=0.5){
    Data1[i,]$outcomes<-1
  }


}
##export data to csv
write.csv(Data,file="C://Users//24732//Desktop//B//Data.csv",quote=F,row.names
= T)
write.csv(Data1,file="C://Users//24732//Desktop//B//Data1.csv",quote=F,row.name
s = T)


################################################
```

```
##problem 3


library(readr)

problem_3 <- read_csv("C:/Users/24732/Desktop/B/problem 3.csv")

Data<-subset(problem_3,select =
c(task,price,`7km_renshu`,`7km_average`,pairs,task_pair_count,outcomes))

Data<-subset(Data,select=-c(pairs))

# ##compute real renshu in area

# for (i in 1:835){

#   Data[i,]$`7km_renshu`<-Data[i,]$`7km_renshu`*Data[i,]$task_pair_count

# }


##use xgboost

Test <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data)

Data$pred_xgb <- predict(xgb,Test[,-1])




##export data to csv

write.csv(Data,file="C://Users//24732//Desktop//B//Data_in_problem_3.csv",quote
=F,row.names = T)
```

```
##################################################

##problem 4

library(readr)

problem_4 <- read_csv("C:/Users/24732/Desktop/B/problem_4_Phase_I.csv")

Data2<-subset(problem_4,select =
c(task,price,`7km_renshu`,`7km_average`,outcomes))


Test <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data2)

Data2$pred_xgb <- predict(xgb,Test[,-1])


for (i in 1:2066){

  if(Data2[i,]$pred_xgb>=0.5){

    Data2[i,]$outcomes<-1

  }

}


write.csv(Data2,file="C://Users//24732//Desktop//B//Data_problem_4_Phase_I.csv"
,quote=F,row.names = T)




##construct Data1
```

```
Data1<- Data2[order(-Data2$pred_xgb),]


##decrease price of pred_xgb>0.5
i=1
total=0
while(i<2067){
  #record price
  price<-Data1[i,]$price
  ##decline 0.5 price
  while(Data1[i,]$pred_xgb>0.7){
    Data1[i,]$price<-Data1[i,]$price-0.5
    #test the try of adjust price
    Test <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data1)
    pred_xgb <- predict(xgb,Test[,-1])
    if(pred_xgb[i]<0.7|Data1[i,]$price<65|price-Data1[i,]$price>5) {
      Data1$price[i]<-Data1$price[i]+0.5
      total<-total+price-Data1$price[i]
      break
    }
    Data1[i,]$pred_xgb<-pred_xgb[i]
  }
  i<-i+1
}


##add price for pred_xgb<0.5
```

```
#first decrease to 65
for (i in 1:2066){
  if(Data1[i,]$pred_xgb<0.5){
    total<-total+Data1[i,]$price-65
    Data1[i,]$price<-65
  }
}



#then incease as for the largest pred_xgb's price
i=1
while(i<2067){
  ##add 0.5 price each turn
  while(Data1[i,]$pred_xgb<0.5){
    Data1[i,]$price<-Data1[i,]$price+0.5
    #test the try of adjust price
    Test <- model.matrix(outcomes~price+`7km_renshu`+`7km_average`,Data1)
    pred_xgb <- predict(xgb,Test[,-1])
    Data1[i,]$pred_xgb<-min(pred_xgb[i]*1)
    if(Data1[i,]$pred_xgb>=0.5|Data1[i,]$price>=85) {
      total<-total+65-Data1$price[i]
      break
    }
  }
```

```
  if (Data1[i,]$pred_xgb<0.5){
    total<-total+Data1[i,]$price-65
    Data1[i,]$price<-65

  }
  if (total<0) {
    total<-total-65+Data1[i,]$price
    Data1[i,]$price<-65
    break
  }
  i<-i+1
}



for (i in 1:2066){
  if(Data1[i,]$pred_xgb>=0.5){
    Data1[i,]$outcomes<-1
  }
}


##export to Data2_final price
write.csv(Data1,file="C://Users//24732//Desktop//B//Data_problem_4_Phase_II.csv
.csv",quote=F,row.names = T)
```

10.2 表格

10.2.1 标价价格概率分布表

价格\区间	中位数	[0,19]	[20,39]	[40,59]	[60,79]	[80,99]	[100,119]	[120,139]	[140,159]	[160,179]	[180,199]
65	101	0	0.04712	0.062937	0.069444	0.134328	0.175	0.142857	0.272727	0.2	0.666667
65.5	74	0	0.115183	0.223776	0.416667	0.358209	0.35	0.333333	0.272727	0.533333	0.333333
66	60	0.024631	0.08377	0.20979	0.208333	0.19403	0.175	0.285714	0.363636	0.2	0
66.5	59	0.034483	0.04712	0.125874	0.111111	0.149254	0.15	0.142857	0.090909	0	0
67	52.5	0.009852	0.041885	0.083916	0.069444	0.074627	0.125	0.047619	0	0	0
67.5	39	0.024631	0.041885	0.041958	0.013889	0.014925	0	0.047619	0	0.066667	0
68	34.5	0.049261	0.062827	0.027972	0.041667	0	0.025	0	0	0	0
68.5	40	0.004926	0.020942	0.041958	0	0	0	0	0	0	0
69	39	0.009852	0.052356	0.020979	0	0.059701	0	0	0	0	0
69.5	29.5	0.014778	0.020942	0.006993	0	0	0	0	0	0	0
70	19	0.246305	0.198953	0.041958	0.027778	0	0	0	0	0	0
70.5	19	0.029557	0.020942	0	0.013889	0	0	0	0	0	0
71	30	0.004926	0.015707	0	0	0	0	0	0	0	0
71.5	22	0.009852	0.015707	0	0	0	0	0	0	0	0
72	24	0.103448	0.115183	0.104895	0.013889	0.014925	0	0	0	0	0
72.5	16	0.034483	0.005236	0.006993	0	0	0	0	0	0	0
73	13	0.039409	0.005236	0	0.013889	0	0	0	0	0	0
73.5	8	0.019704	0.005236	0	0	0	0	0	0	0	0
74	8	0.024631	0	0	0	0	0	0	0	0	0
74.5	8	0.009852	0	0	0	0	0	0	0	0	0
75	8	0.305419	0.08377	0	0	0	0	0	0	0	0

10.2.2 卡方检验图（K=25）

	0.00231	0.002481	0.040136	0.004048	1.119699	0.031951	0.233682	0.016708	0.017868	0.028313	
	0.006802	0.007307	0.003023	0.235812	1.652141	0.005224	1.341549	0.404807	0.299942	0.37097	
	0.011907	0.057876	0.010148	0.235111	0.105619	0.200835	0.222666	0.048692	0.057418	0.267228	
	0.088196	2.99E-08	0.041538	0.021718	0.097067	0.004834	0.004764	0.016879	0.016337	0.010493	
	5.29E-05	0.017267	0.002224	0.000171	0.001814	0.011318	0.009552	0.008874	0.000812	0.001847	
	0.027582	0.000296	0.011375	0.02585	0.015764	9.97E-05	0.028736	0.014771	0.019144	0.006163	
	0.033021	0.027431	0.017143	0.057867	7.77E-05	0.021168	0.0355	0.001029	0.013201	0.026622	
	0.0265	0.028468	0.011961	0.001204	0.045451	0.013051	0.031185	0.055134	0.042678	0.033173	
	0.005602	0.029629	0.01313	0.010344	0.015764	0.001391	0.046295	0.052972	0.00292	0.031872	
	0.015721	0.043327	0.011187	0.047091	0.033624	0.034487	0.025754	0.036226	0.028042	0.021796	
	0.032516	1.432931	0.500871	0.809583	0.080208	0.024287	0.001493	6.03E-05	0.018425	0.014321	
	0.061384	0.023392	0.006926	0.043123	0.002906	0.038762	0.030564	0.005367	0.018425	0.014321	
	0.015038	0.042469	0.050839	0.04574	0.034982	0.060186	0.047456	0.036958	0.028608	0.022236	
	0.054443	0.034858	0.044989	0.017354	0.055804	0.043704	0.03446	0.026837	0.020774	0.016147	
	0.032882	0.012532	0.085328	0.124279	0.003501	0.300731	0.002836	0.009446	0.004025	0.017492	
	0.042225	0.03058	0.008614	0.035919	0.043897	0.012212	0.027107	0.021111	0.016341	0.012702	
	0.029196	0.022915	0.01336	0.029607	0.038933	0.030491	0.024042	0.002069	0.014493	0.011265	
	0.04434	0.038978	0.05966	0.020829	0.031876	0.024964	0.019684	0.015329	0.011866	0.009223	
	0.04434	0.059282	0.017038	0.042869	0.031876	0.024964	0.019684	0.015329	0.011866	0.009223	
	0.067418	0.059282	0.05966	0.042869	0.031876	0.024964	0.019684	0.015329	0.011866	0.009223	
	2.421812	0.290948	0.02365	0.233214	0.010305	0.024964	0.019684	0.015329	0.011866	0.009223	
	3.06329	2.262249	1.0328	2.084602	3.453184	0.93459	2.226377	0.819256	0.666917	0.943851	
数量	66	75	72	77	80	72	84	80	77	112	795
卡方值	202.1771	169.6687	74.36159	160.5144	276.2547	67.29049	187.0157	65.54051	51.35259	105.7113	
	1359.887										df=785

10.2.3 任务打包组合结果表

任务数	价格	平均用户密度	平均信誉值	任务编号					
5	67	30	71500.32756	A0537	A0618	A0701	A0743	A0753	
5	65.5	69	61311.89184	A0001	A0003	A0008	A0358	A0360	
6	65.5	102	91698.65088	A0193	A0327	A0348	A0372	A0383	A0461
5	65	68	56888.94082	A0170	A0312	A0318	A0323	A0430	
6	65	56	23312.748	A0521	A0638	A0754	A0770	A0807	A0834
2	69.5	8	9796.11545	A0496	A0763				
5	68	34	9898.8669	A0527	A0646	A0648	A0651	A0653	
4	65	59	6219.545225	A0484	A0490	A0582	A0588		
2	66	62	5407.7984	A0085	A0089				
4	66	37	3877.785575	A0043	A0065	A0077	A0121		
4	67	11	275.771425	A0538	A0666	A0668	A0676		
2	70	8	233.3732	A0292	A0299				
4	65	82	62067.10175	A0005	A0007	A0028	A0033		
5	65	61	16459.00392	A0265	A0376	A0402	A0408	A0412	
2	70	14	501.81	A0212	A0221				
3	66	32	15472.276	A0141	A0151	A0155			
3	65.5	75	60352.55133	A0109	A0119	A0123			
2	65.5	114	12742.95625	A0051	A0060				
2	66	118	11883.5562	A0040	A0075				
2	65	58	6258.8289	A0597	A0602				
3	65.5	65	39847.29767	A0366	A0368	A0427			
6	65	129	57731.41815	A0132	A0135	A0144	A0183	A0191	A0197
2	68	74	41274.4531	A0010	A0104				
3	67.5	51	23044.7588	A0530	A0780	A0828			
2	65.5	58	5702.09895	A0572	A0577				
2	69.5	7	204.9456	A0584	A0655				
2	72	19	1139.76965	A0707	A0806				
2	66	56	6238.1635	A0603	A0605				
5	65	188	34566.07308	A0181	A0184	A0186	A0382	A0389	
2	69	41	22382.7242	A0519	A0647				
4	65.5	175	30878.45718	A0180	A0218	A0354	A0378		
3	65.5	126	14158.2655	A0336	A0352	A0422			
2	65.5	76	60502.53535	A0011	A0016				
2	65.5	72	41013.02395	A0057	A0112				
4	65	103	88527.9788	A0163	A0309	A0317	A0320		
4	65	114	24899.58423	A0175	A0314	A0316	A0332		
3	65.5	139	88310.33113	A0143	A0195	A0347			
2	65	96	24385.3593	A0131	A0134				
2	66.5	52	22963.2126	A0526	A0759				
3	65	180	22870.74637	A0248	A0250	A0463			
2	65	99	24505.71635	A0142	A0474				
2	65.5	177	28790.707	A0244	A0379				

10.2.4 新项目定价分布表

价格\区间	中位数	9.5	29.5	49.5	69.5	89.5	109.5	129.5	149.5	169.5	189.5
65	101	0.072416	0.949559	3.404165	8.771295	72.97613	89.81944	108.849	78.40172	87.1553	40.34263
65.5	74	0.213241	2.796148	10.02417	25.82865	62.18616	30.50228	36.96463	26.62487	29.59755	13.70018
66	60	0.373315	4.895139	17.54905	21.14671	35.5213	17.42318	21.11453	15.20837	16.90639	7.825665
66.5	59	0.38855	5.094913	18.26524	20.31754	34.12849	16.74001	20.28662	14.61204	16.24348	7.518816
67	52.5	0.503922	6.607746	23.68874	15.66587	26.31482	12.90741	15.64203	11.26664	12.52456	5.797395
67.5	39	0.864734	11.33894	17.54905	9.129258	15.33492	7.52177	9.115365	6.565613	7.298666	3.378422
68	34.5	1.035275	13.57517	14.6582	7.625397	12.8088	6.28271	7.613793	5.484061	6.096359	2.821895
68.5	40	0.830827	10.89433	18.26524	9.50183	15.96075	7.828739	9.48737	6.83356	7.596531	3.516298
69	39	0.864734	11.33894	17.54905	9.129258	15.33492	7.52177	9.115365	6.565613	7.298666	3.378422
69.5	29.5	1.264487	16.58075	12.00112	6.243147	10.48696	5.143848	6.233646	4.489969	4.991276	2.310372
70	19	1.924501	10.89433	7.885295	4.10204	6.890423	3.379749	4.095798	2.95012	3.279502	1.518023
70.5	19	1.924501	10.89433	7.885295	4.10204	6.890423	3.379749	4.095798	2.95012	3.279502	1.518023
71	30	1.239449	16.25243	12.24355	6.369267	10.69881	5.247761	6.359574	4.580672	5.092107	2.357045
71.5	22	1.706879	12.28332	8.890645	4.625037	7.76893	3.810656	4.617999	3.326251	3.697628	1.711566
72	24	1.575648	13.30637	9.631121	5.010243	8.415981	4.128035	5.002618	3.603285	4.005593	1.854117
72.5	16	2.169869	9.662405	6.993629	3.638183	6.111257	2.997568	3.632647	2.616522	2.908658	1.346365
73	13	2.44652	8.569785	6.202793	3.226779	5.420199	2.658605	3.221868	2.320646	2.579748	1.194119
73.5	8	2.650283	7.016346	5.078417	2.641863	4.437683	2.176681	2.637843	1.899985	2.112119	0.977662
74	8	2.650283	7.016346	5.078417	2.641863	4.437683	2.176681	2.637843	1.899985	2.112119	0.977662
74.5	8	2.650283	7.016346	5.078417	2.641863	4.437683	2.176681	2.637843	1.899985	2.112119	0.977662
75	8	2.650283	7.016346	5.078417	2.641863	4.437683	2.176681	2.637843	1.899985	2.112119	0.977662