队伍编号	**205411**
题号	**A**

基于 logistic 回归和 DEA 模型对无车承运平台线路定价问题的优化和评价

摘 要

无车承运平台的出现是当代发展的流行趋势，它降低了平台劳务成本且节约了承运人与司机对接的时间成本。

针对问题一，本问通过查阅文献和资料初步筛选指标，然后进行指标的二次筛选，即将选取的指标和线路指导价做多元线性回归分析，去除线性无关的指标。并作主成分分析来对指标进行降维，得到的主要因素归纳为五大主成分，第一主成分-总里程和线路总成本，第二出成分-成交对象和需求类型 2，第三主成分-打包类型和执行时间，第四主成分-车辆长度和业务类型，第五主成分-细化卸货等待时间和需求类型 1。

针对问题二，本问基于问题一中预处理完的数据，根据查阅文献，增减与线路成交定价相关和不相关的指标，采用数据包络评价模型对线路成交定价进行分析。得出平台整体成交定价不太合理，在成本和交易时间方面没有彰显优势，应该对平台的定价策略进行改进。

针对问题三，本问将问题二中处理完的数据集继续进行数据预处理，基于随机梯度下降法求解线路总成本和线路指导价格的二项逻辑斯特回归模型，由于附件 1 中的调价比例符合多峰分布，基于多峰分布的分布模型生成新的调价比例来调整剩下两次报价，基于问题二中的评价模型，得出的三次报价结果和线路总成本见附件三，通过数据包络模型进行评价，得出平台的部分定价策略合理，资源配置利用程度高。

针对问题四，通过前三问的定价影响因素分析、定价模型、定价评价和调价策略我们对于平台提出的建议是：1、优化货运线路指导价；2、施行差异化线路调价；3、推动建立 C 类承运方联盟。

关键词：梯度下降法；logistic 回归；DEA 模型；主成分分析；分支筛选；多元线性回归

目录

一、问题重述

1.1 问题背景

在传统的货物运输过程中，由于各个货运企业之间并没有积极的进行信息交流，导致整个市场的信息不完善，无序竞争过于激烈。这为物流平台的发展带来了大量的用户基础。同时，随着社会经济的发展，货运需求呈现多元化、个性化的趋势，也使得货运市场对物流企业的要求越来越高，推动了“无车承运人”模式的发展。由于“互联网+”发展环境的影响和国家政策的号召，交通运输部门在运输市场中引入了“无车承运人”的新型业务发展模式。

国家交通运输部办公厅于 2016 年 9 月印发《关于推进改革试点加快无车承运物流创新发展的意见》，并初步公布了 48 个无车承运人试点平台。随着我国无车承运行业的逐步兴起，承运线路的科学定价问题是众多无车承运人平台亟待解决的问题。

1.2 目标任务

问题一：通过定量分析的方法，研究影响无车承运人平台进行货运线路定价的主要因素有哪些，并说明理由。

问题二：根据附件 1 数据，通过建立数学模型，对已经成交货运线路历史交易数据中的定价进行评价。

问题三：建立关于线路定价的数学模型，给出附件 2 的线路任务的三次报价以及总成本定价，并填充在附件 3 的表格中；给出你们的调价策略；评价你们对附件 2 的线路任务所给出的定价。其中附件 3 的表格以 Excel 文件形式，连同论文答卷一起上传至参赛系统，请勿改变附件 3 中各任务 ID 的原有顺序。附件 3 将用于测试报价的准确性，对于某个确定的任务，三次报价中有一次成交，则后续价格将不再考虑。

问题四：根据你们的研究，给无车承运人平台写一封不超过一页的建议信。

二、问题分析

针对问题一，首先通过查阅文献和资料初步筛选指标，再用多重插值法对缺失数据进行补充，其次，结合题目的影响因素的分析，通过多元线性回归，分析线路定价的各个影响关系，做相关性分析和主成分分析。

针对问题二，首先基于问题一中预处理完的数据，重新筛选指标，通过数据包络评价模型对线路成交定价进行分析。

针对问题三，首先，通过指标的选取与相关性分析，对附件二中的数据集，基于随机梯度下降法求解二项逻辑斯特回归模型，其次，结合题目定价模型和调价策略的要求，根据得

出调价比例的数据符合多峰分布，利用 Python 生成符合多峰分布的调价比例进行定价调整，最后，对定价策略通过 MATLAB 进行可视化分析和根据数据包络模型进行评价。

针对问题四，通过前三问的定价影响因素分析、定价模型、定价评价和调价策略，给平台提出建议：1、优化货运线路指导价；2、施行差异化线路调价；3、推动建立 C 类承运方联盟。

三、模型假设

1、假设所选指标能够代表所要研究对象的整体情况。

2、假设变量间的相关性不会对回归造成影响。

3、假设主成分舍弃的成分不会对评价结果造成影响。

4、假设相对重要性可量化。

四、符号说明

序号	变量	变量说明
1	θ^j	$j=1,2,\ldots,N$,第 j 个成交单的评价结果
2	α^j	$j=1,2,\ldots,N$;第 j 个成交单的权重
3	μ_o	$o=1,2,\ldots,m$;第o个投入指标的权重
4	γ_r	$r=1,2,\ldots,n$;第r个产出指标的权重
5	N	成交单的数量
6	m	投入指标的个数
7	n	产出指标的个数
8	X_{oj}	$j=1,2,\ldots,N;o=1,2,\ldots,m$;第 j 个成交单的第 o 个投入指标值
9	X_{oo}	$o=1,2,\ldots,m$;被评价单元的第o个投入指标值
10	y_{ro}	$r=1,2,\ldots,n$被评价单元的第 r 个产出指标值
11	y_{rj}	$j=1,2,\ldots,N;r=1,2,\ldots,n$; 第 j 个成交单的第 r 个产出指标值

五、模型建立和求解

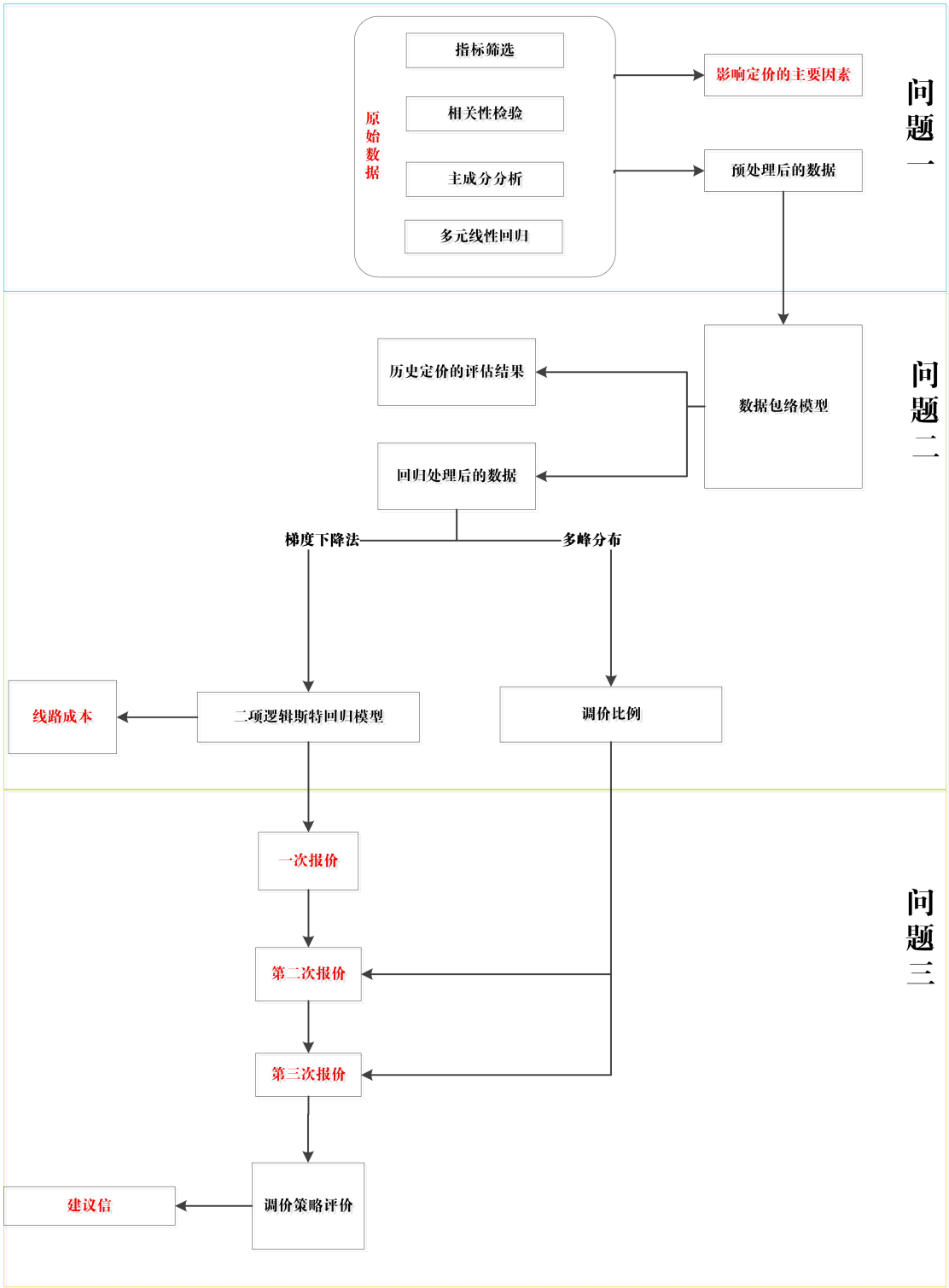

图 5 问题分析流程图

5.1 问题一的分析和建模

5.1.1 成分初步筛选

首先我们利用计划到达时间和计划出发时间两个指标来计算计划时间长：

计算时间长=计划到达世界-计划出发时间 （1）

根据附件 1 中字段释义，可以计算出执行时间：

执行时间=线路编码-始发网点-目的网点 （2）

朱又亮[8]在总结国外铁路客运列车线路使用收费定价方法的基础上，认为线路等级、总里程、需求状况、成交对象一级计划时间长作为影响定价的主要因素，因此，本文选取总里程、业务类型、需求类型 1、需求类型 2、执行时间、成交对象、车辆长度、车辆吨位、打包类型、运输等级、计划卸货时长、计划时间长、线路总成本、需求紧急程度、线路指导价格这些指标来作为数据集进行分析。

由于业务类型、需求类型 1、需求类型 2、成交对象、打包类型、运输等级和需求紧急程度都为文字性重复型指标，因此我们建立一个标准将其量化，具体为表示为：

1、在业务类型中，重货用数值 2 表示，速运用数值 1 表示；

2、在需求类型 1 中，普通用数值 1 表示，区域发货用数值 2 表示；

3、在需求类型 2 中，计划用数值 1 表示，临时用数值 2 表示；

4、在成交对象中，B 即承运商用数值 1 表示，C 即个体司机用数值 2 表示；

5、在运输等级中，一级运输用数值 1 表示，二级运输用数值 2 表示，三级运输用数值 3 表示；

6、在打包类型中，单边用数值 1 表示，周期流向用数值 2 表示，人工用数值 3 表示，周期往返用数值 4 表示；

7、在需求紧急程度中，常规订单用数值 1 表示，紧急订单用数值 2 表示，特急订单用数值 3 表示；

8、将量化后的数据放入数据集，继续进行分析。

5.1.2 缺失值处理

由于数据的缺失，本文采用多重插值算法对数据进行填充。

多重填补法（MI）是由 1987 年美国哈佛大学教授 Rubin 教授在论文中提出，它的思想来源于贝叶斯，认为填补的过程是随机的，它的值来至于已经观测到的值。具体的操作方法上表现就是首先估计出待填补的值,然后加上插补值的不确定性,形成了多组可以选填的值，根据选择填充的依据，选取合适的填补值。它吸收了 EM 法填补的优点，而克服了单一填充的缺点，使得多重填充填补后的数据更加接近真实值。通过 EM 算法我们能够估计出一个比较稳定的参数，并填补一次。下一步要通过数据扩增法来体现插补的不确定性，也就是通过马尔科夫链蒙特卡罗方法（MCMC）根据 EM 求出的一个初值，通过初值建立一条马氏链，从

$p(Y_{mis}|Y_{obs})$的后验分布中模拟参数来对缺失值进行填补。

具体为在随机缺失假设的基础上，就是在X_{obs}基础上X_{mis}是随机缺失的。这样就可以从条件分布$f=(X_{mis}/X_{obs})$中产生填补值$X^{(1)},X^{(2)},\cdots X^{(n)}$。数据的插补是多重填补过程中关键的一步，因为要考虑到完全变量与不完全变量的关系，对每一个缺失值，MI 过程要构造m个填充值，见图 5.1：

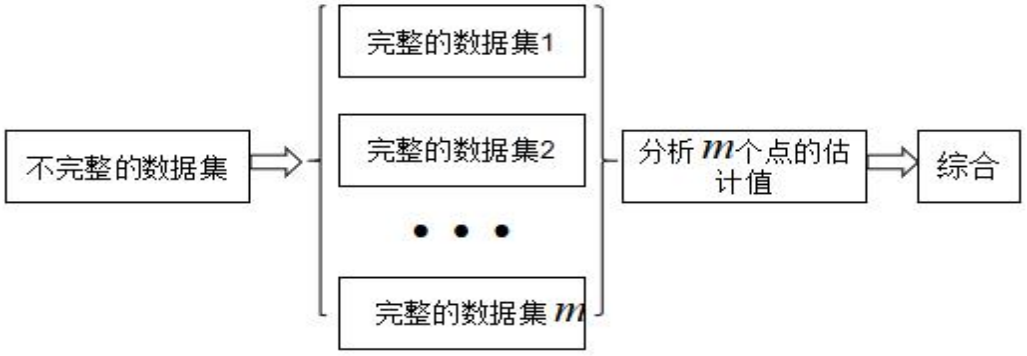

图 5.1 MI 法过程示意图

这m个数据按照一定的顺序或者要求进行排列，那么用于填补缺失数据的数据集会产生一个完整的数据集，通过m次的插补，最终会产生m个完整的数据集。之后对每一个填补后得到的完整数据将会进行分析。一般这些过程没有考虑原数据集中的观测数据和未观测到的数据之间的差别。在分析的基础之上对填补后的数据进行综合分析，即产生最终的统计推断。多重填补法采用的效率主要由m的大小和γ确定的，而γ的值由缺失数据情况而定。

当数据有缺失时，观测数据的后验概率$p(\theta|Y_{obs})$是不可触及也不能进行模拟的，但是只要Y_{obs}被缺失数据Y_{mis}的模拟值扩充，完整数据的后验概率$p(\theta|Y_{obs},Y_{obs})$比较好模拟了。一般假设完整数据满足多元正态混合模型，均值和协方差的 Jeffery 先验分布是：

$$p(\mu,\Sigma)\propto|\Sigma|^{-(k+1)/2} \quad (3)$$

从$\theta=(\mu,\Sigma)$的后验分布产生数据，产生的插补值由数据扩增法（DA）来实现，

$$p(\mu,\Sigma|Y_{obs})\propto|\Sigma|^{-(k+1)/2}\,L(\mu,\Sigma|Y_{obs}) \quad (4)$$

数据扩增法（DA）算法主要分为I步和P步：

（1）I步:通过 EM 算法估计出参数，对独立每一个观测值进行模拟，从给定的Y_{obs}中

条件分布 $p(Y_{mis}\left|Y_{obs},\theta^{(t)}\right.)$ 中估计 $Y_{mis}^{(t+1)}$ 的值

（2）P 步：从上一步得到的完整数据集模拟后验均值和协方差，从 $p(\theta\left|Y_{obs},Y_{mis}^{(t+1)}\right.)$ 中得到 $\theta^{(t+1)}$ 。

重复 I 和 P 步，得到了一条马尔科夫链 $\left(\left\{Y^{(1)},\theta^{1}\right\},\left\{Y^{(2)},\theta^{2}\right\},\cdots\left\{Y^{(t+1)},\theta^{(t+1)}\right\}\right)$，收敛分布到 $p(Y_{mis},\theta\left|Y_{obs}\right.)$，假设迭代收敛到一个比较稳定的分布，那么可以看着从分布中得到了缺失值的独立的近似的值。插补值就可以从这条链中得到，一般将会进行 $m-1$，一共得到 m 条马尔科夫链。在得到模拟的 m 条完数据之后，对这 m 组数据计算相关指标并选择统计方法对插补结果进行分析。

本问基于 SPSS 做插值分析，选取变量置入模型中，选择自动模式，迭代后即可获得插补结果。值得注意的是，由于多重插补的默认设置会出现负值，需要设置大于 0 的约束条件。插补结果及模型如下表 5.1 所示。

表 5.1 插补结果

插补方法		完全条件指定
完全条件指定方法迭代		10
因变量	已插补	打包类型,计划卸货等待时长,线路总成本
	未插补（缺失值过多）	
	未插补（无缺失值）	总里程,业务类型,需求类型 1,需求类型 2,线路指导价（不含税）,执行时间,成交对象,车辆长度,车辆吨位,运输等级,计划时长 h,需求紧急程度
插补序列		总里程,业务类型,需求类型 1,需求类型 2,线路指导价（不含税）,执行时间,成交对象,车辆长度,车辆吨位,运输等级,计划时长 h,需求紧急程度,打包类型,线路总成本,计划卸货等待时长

表 5.2 插补模型

	模型			
	类型(T)	效果	缺失值	插补值
打包类型	Logistic 回归		1	1
线路总成本	线性回归		8	8
计划卸货等待时长	线性回归		6244	6244

如表 5.2 所示，SPSS 设定条件插补的迭代次数为 10，插补项包括打包类型、计划卸货等待时长和线路总成本。打包类型的缺失值使用 Logistic 回归方法插补，线路总成本和计

划卸货等待时长使用线性回归方法插补。

5.1.3 相关性分析

将插补完的数据集，以线路指导价为因变量，选取总里程、业务类型、需求类型 1、需求类型 2、执行时间、成交对象、车辆长度、车辆吨位、打包类型、运输等级、计划卸货等待时长、计划时长、线路总成本和需求紧急程度为自变量输入 SPSS 中做相关分析，相关分析（correlation analysis）是研究现象之间是否存在某种依存关系，并对具体有依存关系的现象探讨其相关方向以及相关程度，是研究随机变量之间的相关关系的一种统计方法。

研究用相关系数r来描述两个变量间线性关系的程度。

$|r| \geqslant 0.95$ 存在显著性相关；

$|r| \geqslant 0.8$ 高度相关；

$0.5 \leqslant |r| \leqslant 0.8$ 中度相关；

$0.3 \leqslant |r| \leqslant 0.5$ 低度相关；

$0 \leqslant |r| \leqslant 0.3$ 关系极弱，认为不相关

pearson 简单相关系数

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})\sum_{i=1}^{n}(y_i - \bar{y})}} = \frac{1}{n}\left(\frac{x_i - \bar{x}}{S_x}\right)\left(\frac{y_i - \bar{y}}{S_y}\right) \quad (5)$$

为了避免单位的不同引起的误差，首先将所给数据进行无量纲化处理，然后对处理后的数据运用 SPSS 统计软件进行双变量相关性分析，分别算出各指标和定价的 pearson 系数。插补后数据，各因素与线路指导价（不含税）的相关性检验结果如表 5.3：

表 5.3 各因素相关性检验

因素	Pearson 相关性	显著性 （双尾）
总里程	.991**	0.000
业务类型	-.216**	.000
需求类型 1	.000	.953
需求类型 2	.130**	.000
执行时间	-.096**	.000
成交对象	-.002	.767
车辆长度	.765**	0.000
车辆吨位	.813**	0.000
打包类型	-.145**	.000
运输等级	-.648**	0.000

计划卸货等待时长	.033**	.000
计划时长/h	.989**	0.000
线路总成本	.991**	0.000
需求紧急程度	.080**	.000

**. 在置信度（双测）为 0.01 时，相关性是显著的。

根据表 5.3 可知，总里程，计划时长，线路总成本与定价显著性相关；车辆吨位与定价高度相关；车辆长度，运输等级，需求紧急程度与定价中度相关；计划卸货等待时长与定价低度相关；业务类型，需求类型 1，需求类型 2，执行时间，成交对象，打包类型与定价关系极弱，认为不相关。

5.1.4 主成分分析

将数据集输入 SPSS 中继续做主成分分析进行降维处理，具体操作为：

（1）检验因子分析的可行性

表 5.4 KMO 和巴特利特检验

KMO 取样适切性量数。		.687
Bartlett 的球形度检验	上次读取的卡方	432672.949
	自由度	105
	显著性	.000

表 5.4 的 KMO 检验用于检查变量间的偏相关性，值为 0.699，大于 0.6，说明可以做因子分析。Bartlett's 球形检验用于检验相关阵是否是单位阵， 值为 0，<0.01 说明指标间并非独立，取值是有关系的。因此，可以进行主成分分析。

（2）主成分的确定

表 5.5 总方差解释

组件	初始特征值			提取载荷平方和		
	总计	方差百分比	累积 %	总计	方差百分比	累积 %
1	6.040	40.266	40.266	6.040	40.266	40.266
2	2.037	13.578	53.844	2.037	13.578	53.844
3	1.596	10.639	64.483	1.596	10.639	64.483
4	1.294	8.629	73.111	1.294	8.629	73.111
5	1.103	7.356	80.467	1.103	7.356	80.467
6	.875	5.830	86.298			
7	.699	4.658	90.956			
8	.636	4.238	95.194			
9	.539	3.592	98.787			
10	.141	.939	99.725			
11	.021	.142	99.868			
12	.012	.081	99.948			

13	.005	.034	99.982			
14	.002	.011	99.993			
15	.001	.007	100.000			

提取方法：主成份分析。

如表5.5所示,SPSS总共提取了5个特征值大于1的主成分,特征值分别为6.040,2.037，1.596，1.294，1.103。贡献率分别为40.266%，13.578%，10.639%，8.629%，7.356%。

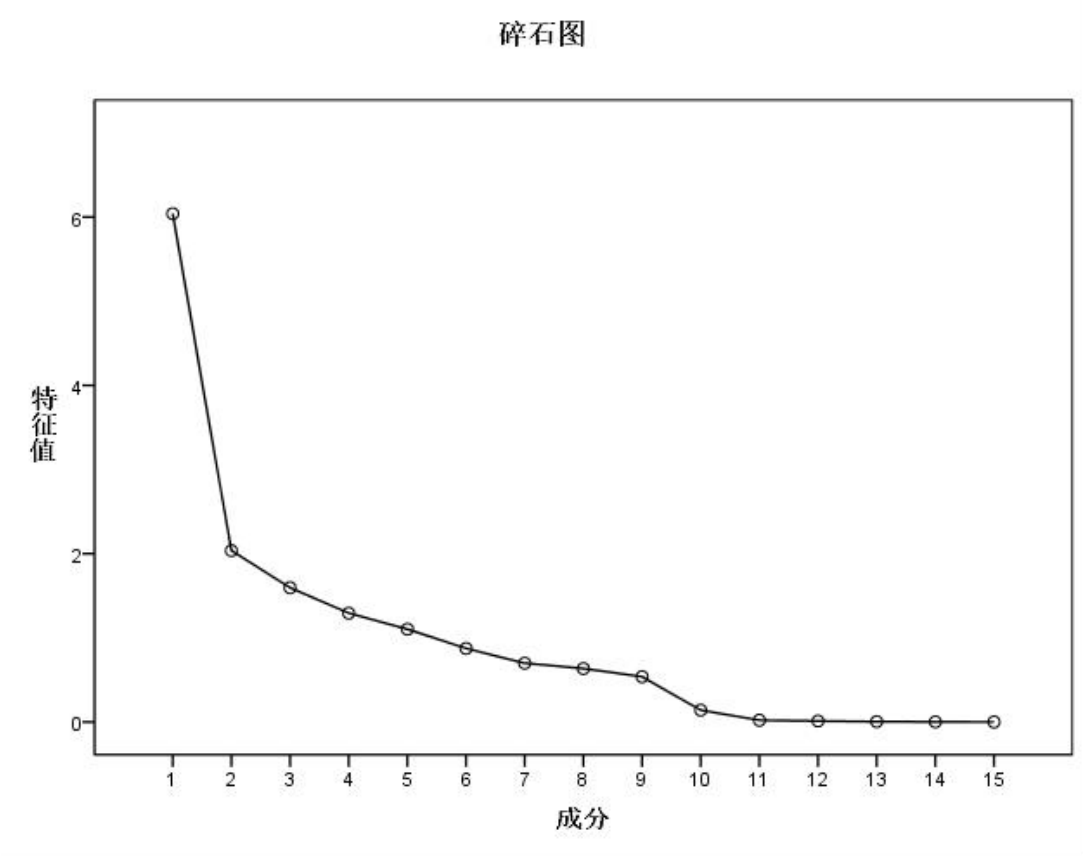

图5.2碎石图

图5.2给出了特征根按大小分布的折线图，可以直观地判定提取的因子，由图可知前5个成分的确呈现高度相关性。

（3）主成分的表达式

5.6 成分矩阵

	组件				
	1	2	3	4	5
总里程	.969	-.161	-.130	-.033	-.049
线路总成本	.968	-.151	-.139	-.028	-.048
线路指导价（不含税）	.965	-.153	-.136	-.028	-.049
计划时长/h	.962	-.166	-.149	-.039	-.054

车辆吨位	.916	.184	.043	.253	.066
车辆长度	.873	.229	-.026	.302	.091
运输等级	-.748	.121	-.618	.084	-.066
需求类型 2	.216	.641	-.057	-.430	-.078
成交对象	.072	.595	.249	.148	-.378
打包类型	-.212	-.582	.258	.301	-.044
需求紧急程度	.165	.563	.046	-.468	.090
执行时间	.066	.130	.967	.047	.053
业务类型	-.134	.577	-.250	.627	.172
需求类型 1	.039	.163	.049	.168	.802
计划卸货等待时长	.025	-.195	-.040	-.433	.491

提取方法：主成份分析。

由表 5.6 可知，总里程、线路总成本在第一成分上载荷较大，即与第一主成分的相关系数较高。成交对象、需求类型 2 与第二主成分的相关系数较高，打包类型、执行时间与第三主成分的相关系数较高，车辆长度、业务类型与第四主成分的相关系数较高；计划卸货等待时长、需求类型 1 与第五主成分的相关系数较高，具体表示为表 5.7。

表 5.7　线路定价影响因素主成分

主成分类型	具体组合
第一种主成分	总里程-线路总成本
第二种主成分	成交对象-需求类型 2
第三种主成分	打包类型-执行时间
第四种主成分	车辆长度-业务类型
第五种主成分	计划卸货等待时长-需求类型 1

（4）定价模型

设定无车承运人平台进行货运线路价格为 y，z_i 为第 i 个主成分，则用统计回归分析获得无车承运人平台进行货运线路价格的定价模型如下：

$$y = 0.40266z_1 + 0.12235z_2 + 0.10504z_3 + 0.08031z_4 + 0.06756z_5 \quad (6)$$

5.2 问题二的分析和建模

5.2.1 数据预处理

首先我们利用问题一中在 SPSS 中使用插值法对实际结束时间这一指标中缺失的数据进行补充，然后计算交易时长和实际时长指标的计算公式表示为：

交易时长 = 交易开始时间 - 交易结束时间　（1）

实际时长 = 实际发车时间 - 实际结束时间 （2）

继续使用问题一处理后的数据集，由于已经成交货运线路的定价，即线路价格（不含税）与指标，它与计划时长和计划卸货时间的关联不大，它和实际时长密切相关，在下面相关性分析得到了验证，因此我们去除处理后数据集的计划卸货时间和计划时长指标以及线路指导价（不含税），又由于本无车承运 人平台在当前阶段较为关注的目标是快速促进成交和较低的承运成本，因此引入线路价格（不含税）、实际时长、交易时长指标。

5.2.2 回归分析

在 SPSS 中选取线路价格（不含税）作为因变量，以总里程、业务类型、需求类型 1、需求类型 2、成交对象、运输等级、实际时长、交易时长、线路总成本作为自变量进行相关性分析，结果如表 5.8 所示：

表 5.8 相关性分析

		总里程	业务类型	需求类型1	需求类型2	成交对象	运输等级	实际时长	交易时长	线路总成本
线路价格（不含税）	Pearson 相关性	.989**	-.205**	-.004	.110**	-.011	-.635**	.542**	.106**	.987**
	显著性（双尾）	.000	.000	.590	.000	.159	.000	.000	.000	.000
	N	16016	16016	16016	16016	16016	16016	16016	16016	16016

**. 在置信度（双测）为 0.01 时，相关性是显著的。

一般来说相关性大于 0.95，我们一般认为是显著性相关。如图 5.8 所示，总里程、线路总成本与定价显著性相关，实际时长、运输等级与定价高度相关；计划卸货等待时长与定价中度相关，业务类型与定价低度相关，需求类型 1、需求类型 2、执行时间、成交对象、交易时长与定价关系极弱，认为不相关。

5.2.3 分支筛选法

由于总体数据决策单元为 16016 个，由于数据量巨大，在模型求解时运行速度相当缓慢。本文在保持原样本数据集特性的基础上，基于分支筛选法来选取 100 组数据作为样本来代替总体。在选用分支筛选法筛选样本的过程中，本文根据业务类型、需求类型 1、需求类型 2、成交对象、运输等级来设置了 5 层筛选，因为这些指标数据划分的类型只有两种或三种，根据上面的指标值进行数值类型的 5 层划分，筛选流程图如图 5.3 所示：

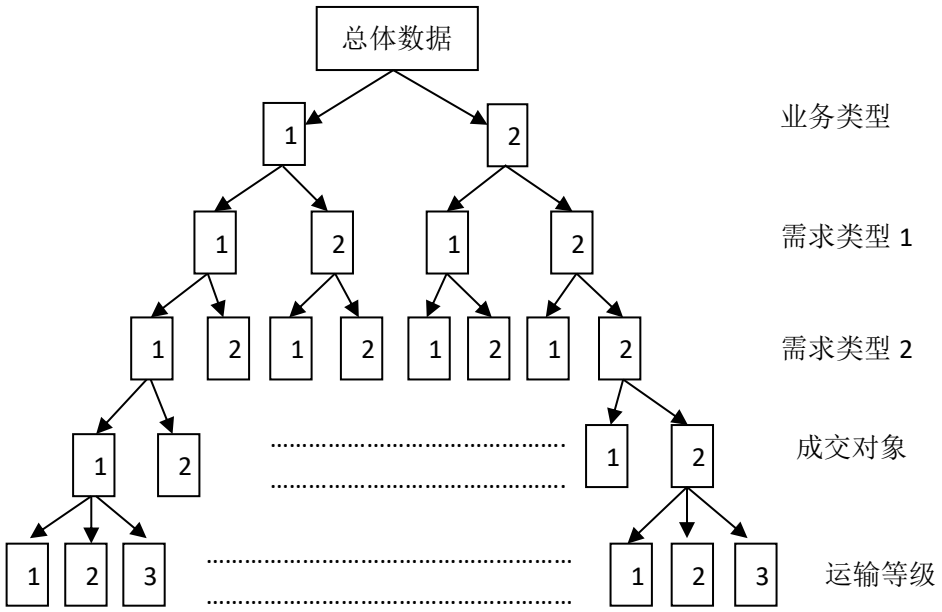

图 5.3 样本筛选流程图

汇总到第五级筛选完后，可以得到 48 种以上筛选指标组合类型，其类型占总体数据集百分比及其在总体数据集选取的个数统计为表 5.9 所示（具体数据请看附件），筛选后的样本数据如附录所示。

指标类型选取个数=100（样本个数）×百分比（3）

表 5.9 指标类型统计表

类型序号	指标类型名称	百分比（%）	选取个数
1	业务类型：1-需求类型 1：1-需求类型 2:1-成交对象：1-运输等级：1	6	6
2	业务类型：1-需求类型 1：1-需求类型 2:1-成交对象：1-运输等级：2	33	33
3	业务类型：1-需求类型 1：1-需求类型 2:1-成交对象：1-运输等级：3	3	3
…	…	…	…

样本选取的个数越多以及筛选数据集分的层次越多，所获得的样本质量和特性就越接近原数据集，但运行速度相对会慢，本文在模型计算 2000 个样本，平均 8s 算出一个决策单元结果，计算 100 个样本，平均 1s 算出一个决策单元的结果，考虑到时间方面，本文采取 100 个样本进行实证分析。

5.2.4 构建评价模型

无车承运人平台线路定价问题就是平台根据路线状况以及花费成本和时间等多角度来确定每一单的价格，然后再按司机意愿程度来分配单量，平台发布竞价信息分别为 B 端和 C

端，用来与司机进行对接，如图 2 所示。对无车承运人平台线路成交货运线路历史交易数据中的定价进行评价就是一个一阶段的过程，也是一个线性优化问题。整个流程的主要对象就是平台，成交时间和线路总成本可以通过平台根据线路总里程和业务类型、需求类型 1、需求类型 2、成交对象、运输等级、实际时长来确定，由于无车承运 人平台在当前阶段较为关注的目标是快速促进成交和较低的承运成本，成交的快慢和承运成本的高低以及线路价格的高低是对成交定价的单的质量进行评价的依据，因此我们选取总里程、业务类型、需求类型 1、需求类型 2、成交对象、运输等级、实际时长、线路价格（不含税）作为投入指标，选取交易时长、线路总成本作为产出指标，图 5.4 为线路成交货运线路历史交易数据中的定价进行评价的过程。

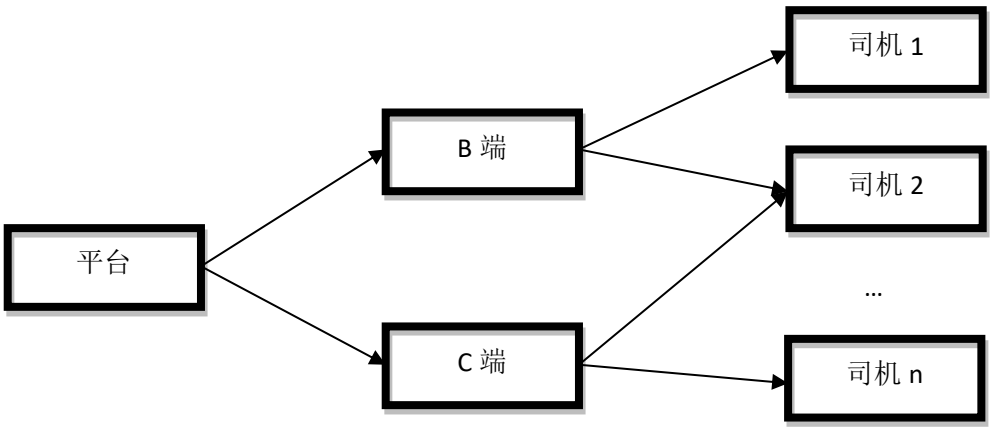

图 5.4 无车承运人平台线路定价问题流程图

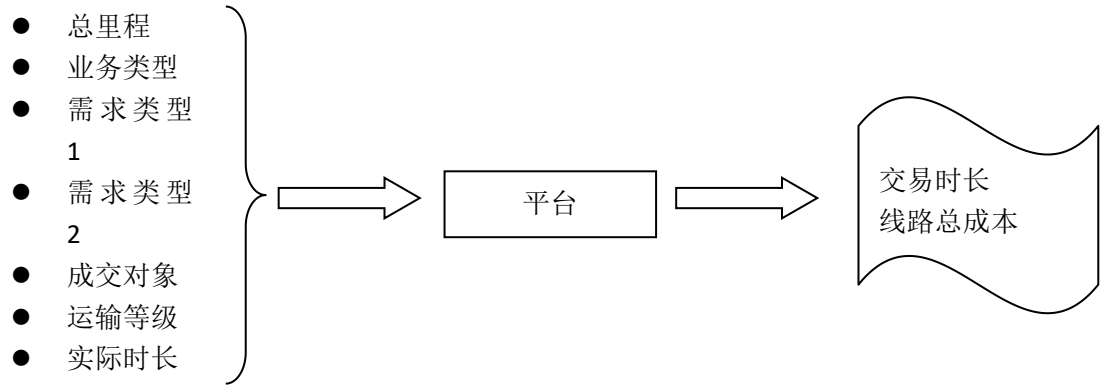

图 5.5 成交定价评价模型框架

数据包络模型是一种客观评价模型，相对于层次分析法更加具有客观性。Zhu J 等人提出了数据包络模型（即 DEA 模型）来做效率评价[7]，本文以普通的 DEA 模型来构建成交货运

线路历史交易数据中的定价评价模型，表 5.10 为变量说明。

表 5.10 变量说明

决策变量	变量说明
θ^{j}	$j=1,2,\cdots,N$,第 j 个成交单的评价结果
$\propto_{j}$	$j=1,2,\cdots,N$;第 j 个成交单的权重
μ_{o}	$o=1,2,\cdots,m$;第o个投入指标的权重
ν_{r}	$r=1,2,\cdots,n$;第r个产出指标的权重
已知变量	
N	成交单的数量
m	投入指标的个数
n	产出指标的个数
x_{oj}	$j=1,2,\cdots,N;o=1,2,\cdots,m$;第 j 个成交单的第 o 个投入指标值
x_{oo}	$o=1,2,\cdots,m$;被评价单元的第o个投入指标值
y_{ro}	$r=1,2,\cdots,n$被评价单元的第 r 个产出指标值
y_{rj}	$j=1,2,\cdots,N;r=1,2,\cdots,r$; 第 j 个成交单的第 r 个产出指标值

模型构建如下所示：

$$\min\theta=\frac{\gamma_{r}y_{ro}}{\mu_{o}x_{oo}} \quad (4)$$

s.t.

$$\sum_{j=1}^{N}x_{oj}\alpha_{j}\ \leq\ \theta_{j}x_{oo},o\ =\ 1,2,\ldots,m \quad (5)$$

$$\sum_{j=1}^{N}y_{rj}\alpha_{j}\ \geq\ y_{ro},r\ =\ 1,2,\ldots,n \quad (6)$$

$$\alpha_{j}\ \geq\ 0,j\ =\ 1,2,\ldots,N \quad (7)$$

5.2.5 模型求解

模型问题是一个线性优化问题，模型求解是在 python 中使用 scipy 进行数值优化，基于一序列最小二乘法来求解线性模型，求得的数据评价结果如表 所示，python 代码见附录，设置输入指标数为 8。求解结果如表 5.11 所示（具体请看附件）：

表 5.11 模型求解结果

任务 id	评价结果
1007	1.00
1020	0.94
1027	1.00
1030	1.00
1032	0.97
...	...

评价结果分布在[0,1]的区间中，评价结果分数越小，说明订单的成交定价越合理，从平台的角度来看就更加有利，其中任务 id 为 4 的订单的评价结果分数最小，其成交价格为 1100，其中存在 38 个订单的评价结果分数为 1，占总体的 38%，这部分的订单成交定价极其不合理。样本数据评价结果分数整体均值为 0.95，说明平台整体成交定价不合理，定价策略需进一步改进。

5.3 问题三的分析和建模

5.3.1 数据处理

基于问题 1 中数据预处理后的数据集，本文引入交易对象指标进入数据集，并对其进行量化，指标交易对象中 B 定义为值 1，C 定义为值 2，BC 定义为值 3，对比附录 2 中的数据指标，选取指标总里程、业务类型、需求类型 1、需求类型 2、执行时间、车辆长度、车辆吨位、打包类型、运输等级、计划卸货时间、计划时间长、交易对象、需求紧急程度、线路总成本形成新的数据集，将其作为训练数据集 1，选取指标总里程、业务类型、需求类型 1、需求类型 2、执行时间、车辆长度、车辆吨位、打包类型、运输等级、计划卸货时间、计划时间长、交易对象、需求紧急程度、线路指导价形成新的数据集，将其作为训练数据集 2。然后分别从训练数据集 1 中选取 20 个决策单元数据作为测试数据集 1，同样从训练数据集 2 中选取 20 个决策单元数据作为测试数据集 2，并对这 2 个训练数据集和 2 个测试数据集中线路总成本和线路指导价格进行归一化处理，处理后值为 0 或 1，将附件 2 中的数据进行问题一中同样的数据预处理，其中它的量化标准具体为：

1、在业务类型中，重货用数值 2 表示，速运用数值 1 表示；

2、在需求类型 1 中，普通用数值 1 表示，区域发货用数值 2 表示，二程接驳用数值 3 表示；

3、在需求类型 2 中，计划用数值 1 表示，临时用数值 2 表示；

4、在成交对象中，B 即承运商用数值 1 表示，C 即个体司机用数值 2 表示；

5、在运输等级中，一级运输用数值 1 表示，二级运输用数值 2 表示，三级运输用数值

3 表示；

6、在打包类型中，单边用数值 1 表示，周期流向用数值 2 表示，人工用数值 3 表示，往返用数值 4 表示；

7、在需求紧急程度中，常规订单用数值 1 表示，紧急订单用数值 2 表示，特急订单用数值 3 表示；

将附件 2 数据预处理完后形成的新数据集，选取指标总里程、业务类型、需求类型 1、需求类型 2、执行时间、车辆长度、车辆吨位、打包类型、运输等级、计划卸货时间、计划时间长、交易对象、需求紧急程度、线路总成本作为拟合数据集 2，选取指标总里程、业务类型、需求类型 1、需求类型 2、执行时间、车辆长度、车辆吨位、打包类型、运输等级、计划卸货时间、计划时间长、交易对象、需求紧急程度、线路第一次定价作为拟合数据集 1。

5.3.2 模型建立

为了获取线路总成本以及线路指导价和其他变量的关系式，本文采用二项逻辑斯特回归模型来对其进行分析，第一组：以总里程、业务类型、需求类型 1、需求类型 2、执行时间、车辆长度、车辆吨位、打包类型、运输等级、计划卸货时间、计划时间长、交易对象、需求紧急程度为自变量，设置线路总成本为因变量。

第二组：以总里程、业务类型、需求类型 1、需求类型 2、执行时间、车辆长度、车辆吨位、打包类型、运输等级、计划卸货时间、计划时间长、交易对象、需求紧急程度为自变量，设置线路指导价为因变量。

（1）二项逻辑回归模型：

$$P(Y=0\left|x\right)=\frac{1}{1+e^{\omega x+b}} \quad (8)$$

$$P(Y=1\left|x\right)=\frac{e^{\omega x+b}}{1+e^{\omega x+b}} \quad (9)$$

其中 $x\in R^n$ 是输入，$Y\in\{0,1\}$ 是输出，$\omega\in R^n$ 和 $b\in R$ 是参数，ω 称为权值向量，b 称为偏置，ωx 称为 ω 和 x 的内积。有时为了方便，将权值向量和输入向量加以扩充，仍然记为 ω 和 x，但是 $\omega=\left(\omega^1,\omega^2,\cdots,\omega^n,b\right)^T$，$x=\left(x^1,x^2,\cdots,x^n,1\right)^T$。在这种情况下，二项逻辑回归模型如下：

$$P(Y=0\left|x\right)=\frac{1}{1+e^{\omega x}} \quad (10)$$

$$P(Y=1\Big|x)=\frac{e^{\omega x}}{1+e^{\omega x}} \quad (11)$$

定义 $sigmoid$ 函数为

$$sigmoid(z)=\frac{1}{1+e^{-z}} \quad (12)$$

（2）似然函数法估计模型参数 ω

设 $P(Y=1|x)=\pi(x)$，$P(Y=0|x)=1-\pi(x)$，则似然函数为 $\Pi\left[\pi(x_i)\right]^{y_i}\left[1-\pi(x_i)\right]^{1-y_i}$

对数似然函数为

$$L(\omega)=\sum_{i=1}^{N}\left[y_i\log\pi(x_i)+(1-y_i)\log(1-\pi(x_i))\right] \quad (13)$$

$$=\sum_{i=1}^{N}\left[y_i(\omega x_i)-\log(1+e^{\omega x_i})\right] \quad (14)$$

加入正则项的损失函数

$$L(\omega)=-\frac{1}{N}\sum_{i=1}^{N}\left[y_i\log\pi(x_i)+(1-y_i)\log(1-\pi(x_i))\right]+\frac{\lambda}{2N}\|\omega\|_2^2 \quad (15)$$

$$=\frac{1}{N}\sum_{i=1}^{N}\left[-y_i(\omega x_i)+\log(1+e^{\omega x_i})\right]+\frac{\lambda}{2N}\|\omega\|_2^2 \quad (16)$$

求 $L(\omega)$ 的极大值，得到 ω 的估计值

不加正则项求 ω

$$\frac{\partial L}{\partial\omega_j}=x_{ij}(-y_i+sigmoid(\omega x_i)) \quad (17)$$

接下来可以由随机梯度下降法求解 ω

同理加入正则项的梯度为

$$\frac{\partial L}{\partial\omega_j}=\frac{1}{N}\left[x_{ij}(-y_i+sigmoid(\omega x_i))+\lambda\omega\right] \quad (18)$$

5.3.3 随机梯度下降法算法实现

对逻辑斯特模型求解，本文采用随机梯度下降法，算法具体流程如下所示：

1、设置变量：

matrix *为读入数据矩阵*
test_matrix *为测试数据矩阵*
y *为分类情况，y[i]表示第 i 组数据的分类情况*
test_y *为测试数据集的分类情况*
x *为特征矩阵，其中 x[i]表示第 i 个实例的特征取值情况,最后一维为 1*
test_x *为测试数据集的特征矩阵*
w *为对应扩充特征后的 w*
n *为特征数的个数，其中 w 是 n+1 维的*
dataSum 为*数据量*
testSum 为*测试数据集大小*

2、生成一个二维数据

满足贝叶斯：协方差矩阵半正定，例如

$$\mathrm{cov} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

不满足贝叶斯：当协方差不等于 0 时，两个参数相关，则不独立，例如，2 维数据均相关，不独立。

$$\mathrm{cov} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

3、读取数据

使用 pandas_read_csv 读取 csv 格式的数据，然后再将读入的 DataFrame 结构使用.valus 转化为 ndarray，然后使用矩阵切片和扩充生产 x.y。

4、随机梯度下降法

自行设置迭代次数 door，每次选取一组数据，根据以下公式进行求解，观察不同迭代次数的收敛情况。

$$\frac{\partial L}{\partial w_j} = x_{ij}(-y_i + \mathrm{sigmoid}(wx_i)) \quad (19)$$

5、计算正确率和权重

计算 w*x 和权重 w 的值，与 0 比较，偌大与或者等于零预测为 1；小于 0 预测为 0。统计预测正确的样本数，计算预测的正确率。

5.3.4 拟合第一次定价和成本定价结果

在 python 中我们通过机器学习运用随机梯度下降法求解逻辑斯特回归模型，代码见附录，计算出来第一组的正确率为 1，各个指标的权重系数分别为-0.35、0.97、0.97、0.97、-0.5、0.844、0.91、0.91、0.94、-0.13、-0.65、0.93、0.97、0.97。逻辑斯特公式表示为：

$$L(w)=\sum_{i=1}^{N}[y_i(wx_i)-\log\ (1+e^{wx_i})] \tag{20}$$

$$w_1=-0.35,w_2=0.97,w_3=0.97,w_4=0.97,w_5=-0.5,w_6=0.844,$$

$$w_7=0.91,w_8=0.91,w_9=0.94,w_{10}=-0.13,w_{11}=-0.65,w_{12}=0.93$$

$$w_{13}=0.97,w_{14}=0.97;$$

同时计算出第二组的正确率为0.94，各个指标的权重系数分别为-0.35、0.97、0.97、0.97、-0.56、0.844、0.91、0.93、0.94、0.1、-0.8、0.97、0.97、0.97。逻辑斯特公式表示为：

$$L(w)=\sum_{i=1}^{N}[y_i(wx_i)-\log\ (1+e^{wx_i})] \tag{21}$$

$$w_1=-0.35,w_2=0.97,w_3=0.97,w_4=0.97,w_5=-0.56,w_6=0.844,$$

$$w_7=0.91,w_8=0.93,w_9=0.94,w_{10}=0.1,w_{11}=-0.8,w_{12}=0.97$$

$$w_{13}=0.97,w_{14}=0.97;$$

将附录2中处理完的拟合数据集1和拟合数据集2分别代入模型（20）和模型（21），求得线路总成本和第一次定价。

5.3.4 调价策略

价格调整策略的制定应用了博弈论的思想。第一次报价必然尽可能低，最好逼近承运方心理下限。如果未成交，第一次调价中要以合理价格为目标，使得承运方在第二次报价下能够盈利。如果还未成交，则第二次调价种要以让承运方接单为目标，即第三次报价应当是所有报价中最高。

（1）步骤一：确定附录1中调价比例的分布

首先，我们对各项指标进行分类，拆分附录一以研究指标的变化对调价策略的影响。参考问题一中关于相关性的分析结果得出业务类型、需求类型、运输等级是影响调价策略的重要因素，于是将数据为拆分为六类。结果如下表5.12所示：

表 5.12 组合因素的符号说明

符号	说明
111	速运+计划+一级运输
112	速运+计划+二级运输
113	速运+计划+三级运输
121	速运+临时+一级运输
122	速运+临时+二级运输
123	速运+临时+三级运输
211	重货+计划+一级运输
212	重货+计划+二级运输
213	重货+计划+三级运输
221	重货+临时+一级运输
222	重货+临时+二级运输
223	重货+临时+三级运输

为了得到分布，我们进行了调价策略与各影响因素的交叉分析，部分结果如图 5.6 所示：

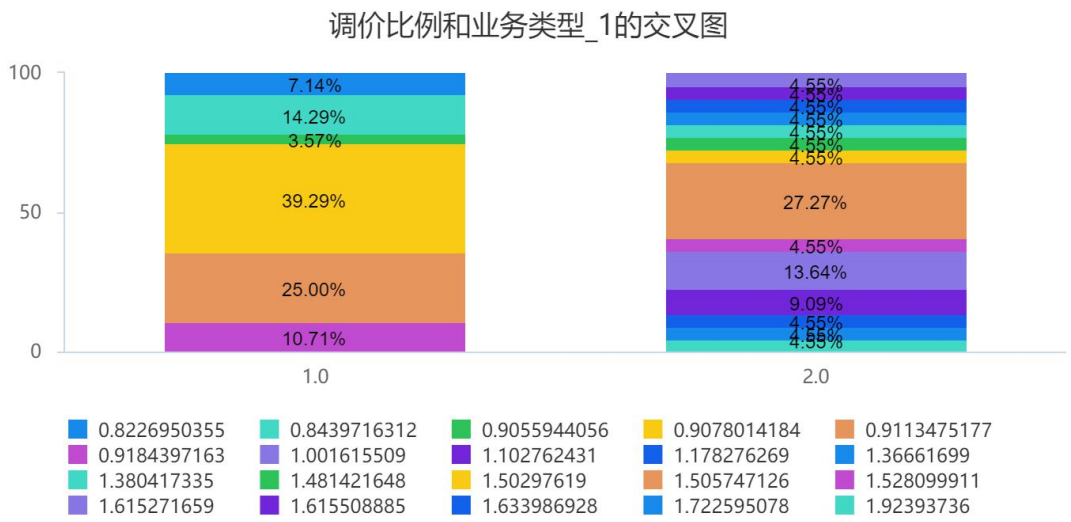

图 5.6 调价比例与业务类型的关系

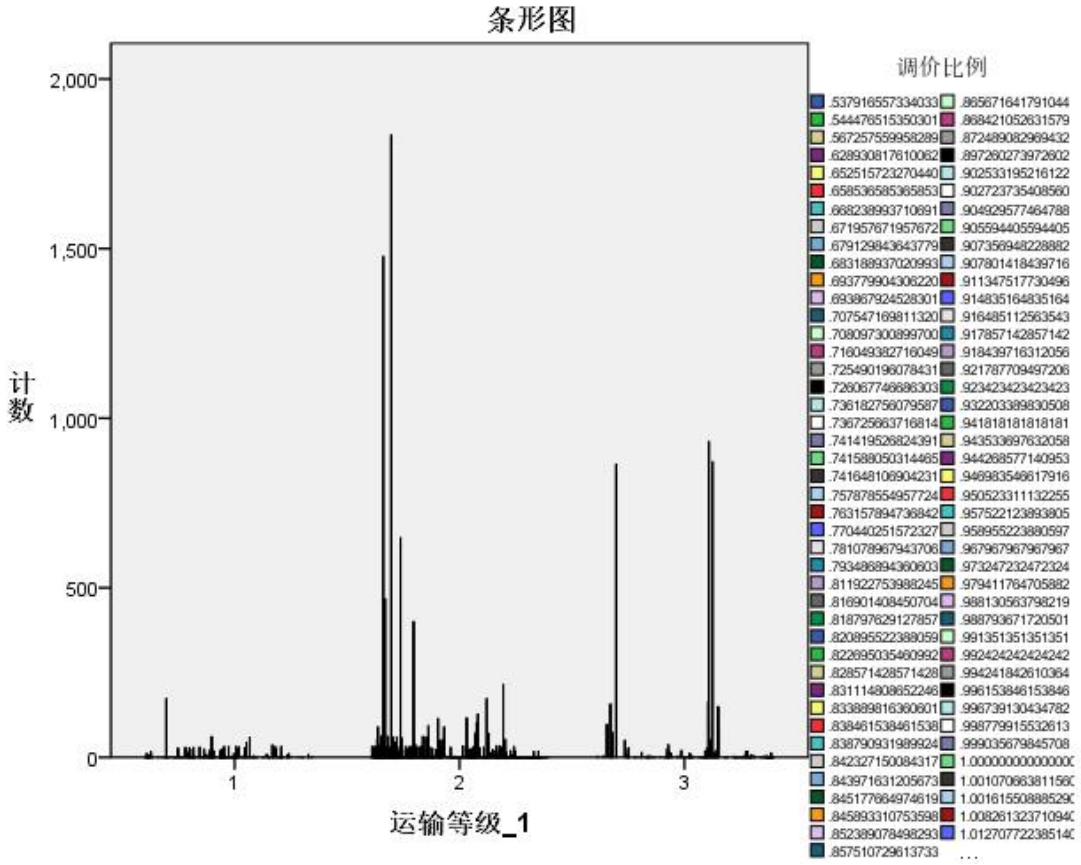

图 5.7 调价比例与运输等级的关系分布

由图 5.7 可知，调价比例与各项因素得关系呈现多峰分布。多峰分布是分布中的多个分数附近集中着较多的次数，以致次数分布曲线有多个隆起的峰，是正态分布的特殊形式。

正态分布（Normal distribution）又名高斯分布（Gaussian distribution），若随机变量 X 服从一个数学期望为 μ、标准方差为 σ 2 的高斯分布，记为：

$$X\sim N(\mu,\sigma 2)$$

则其概率密度函数为

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\delta^2}} \qquad (22)$$

正态分布的期望值 μ 决定了其位置，其标准差 σ 决定了分布的幅度。多峰分布需要根据期望值及标准差进行描绘。韩志国（2010）[10]指出，对于样本数据表现为一大一小两个峰值的多峰分布特征有必要加以识别和测度，为分析产生的原因提供线索。 多峰分布定义如下：定义 ：对于一组观测值分布在[x_1 ， x_2] 值域内的离散样本数据，其中至少有一个观测值 j^*具有最多的样本数目，在 x_1， x_2 的两端至 j*的区间内如果存在 k 个观测值 j^k（ 0 <k <x2 -x1），j^k的样本数目大于其左右相邻观测值的样本数目，即称这组离散样本数据的分

布形态为多峰分布形态。 用 j^k 与 j*之间出现的凹陷区域面积的大小来测度一组样本数据的多峰分布的显著程度，称为多峰分布度，记为 M。设一个样本数量为 n 的离散值样本组，样本数据限定在某区间[x1, x2] 内,其中 n/2 的样本观测值为 x1 ，n/2 的样本观测值为 x2 。显然，该样本数据的分布具有最大的凹陷区域面积，多峰分布度计算式如下：

$$M=\frac{\sum_k m_k}{\frac{n}{2}(x_2-x_1-1)}=\frac{2\sum_k m_k}{n(x_2-x_1-1)},(k=1,2,...,K;x_2>x_1) \tag{23}$$

其中 K 为样本组中凹陷区域的数目 ；m^k 为样本组中第 k 个凹陷区域的面积

（2）步骤二：根据分布生成附录 3 的调价比例

统计 12 类数据的分调价策略的范围，对其最小值及最大值进行了统计，取最大值与最小值的均值作为期望值，12 组组合的正态分布的均值和方差如表 5.13 所示。

表 5.13 各类数据调价比例的范围

类别	最小值	最大值	期望值 μ	方差
111	0.537916557	1.906742936	1.222329747	0.2
112	0.56725756	3.212903226	1.890080393	0.2
113	0.537916557	3.212903226	1.875409892	0.2
121	0.537916557	1.92393736	1.230926959	0.2
122	0.56725756	3.212903226	1.890080393	0.2
123	0.537916557	3.212903226	1.875409892	0.2
211	0.537916557	1.92393736	1.230926959	0.2
212	0.56725756	3.212903226	1.890080393	0.2
213	0.537916557	3.212903226	1.875409892	0.2
221	0.537916557	1.92393736	1.230926959	0.2
222	0.56725756	3.212903226	1.890080393	0.2
223	0.537916557	3.212903226	1.875409892	0.2

通过在 Excel 中基于上述表 中 12 个正态分布参数进行随机数的生成，生成的随机调价策略指标比例如表 5.14 所示。

表 5.14 服从范围及分布限定的部分随机数

111	112	113	121	122	123
1.300655457	1.909542488	2.104365377	1.566977785	1.678506043	2.082610402
1.154838049	2.049741153	1.754475081	0.881331731	2.02742554	2.128624484

1.202521748	1.983327637	1.89379265	0.96220773	2.303938411	1.920145184
1.182295889	1.805193757	1.950186817	1.346990945	1.675027633	1.955677809
1.568120717	2.012354772	2.039740737	1.098655602	1.900287779	1.987381702
1.438149818	1.681574768	2.152326773	1.227110564	1.86048531	1.704049276
1.39128612	1.967748764	1.882451017	1.282291148	1.872755197	2.010319044
0.878639084	1.669605989	2.078377732	1.076556774	1.823352941	1.729772565
1.055825937	1.503595813	2.07405193	1.607762417	1.956377628	1.935427373
1.203841034	2.052280903	1.814819576	1.204694959	1.858203697	2.141438951
1.614109894	1.895207544	2.006109692	1.208465111	2.054719345	1.983645502
1.391582325	2.09408652	1.724100052	1.351045925	2.116983944	2.114713303
1.056178226	1.908929329	1.901165136	1.160920779	2.045872626	1.952280859
1.246469694	2.222467927	2.068431936	1.424352766	1.655840325	1.375617037
1.22351786	2.080358131	1.796773242	1.252623075	1.726250367	2.098540455
1.107396396	1.432692105	1.962136385	1.205415551	2.164174521	2.116449273
1.316682832	1.675644406	2.031310058	1.20991041	1.752850477	1.662996873
0.991102838	2.063648952	1.801741346	1.486203443	2.013056355	2.038729388

（3）步骤三：调整报价

在 Python 中设定范围和分布的调价比例的随机数对附录三中第一次报价的数据进行赋值，所得结果即为第二次报价。

由于第三次报价是最后一次报价，目标是让 c 端或者 B 端的承运方接单，因此调价比例需要相应提高。我们在考虑第 2 次报价效果的前提下，设置第三次报价的结果为第一次报价基础乘以相应的调价比例，均值为调价比例乘以 120%。此后重新调整调价比例，再次生成调价比例并应用 Python 进行赋值，所得结果即为第 3 次报价。

三次报价及线路总成本的部分结果如表 5.15 示例。（详情见计算结果）：

表 5.15 三次报价及线路总成本

编号	任务 id	第一次报价	第二次报价	第三次报价	线路总成本
1	17281	131.71	297.09	356.51	130.81
2	15629	223.48	406.86	488.23	223.73
3	17034	131.67	253.03	303.64	130.67
4	17008	131.67	231.90	278.28	130.67
5	15630	223.18	505.67	606.80	223.46
6	17282	131.71	249.08	298.90	130.82
7	15651	223.80	406.75	488.11	224.06
8	17035	131.65	230.47	276.56	130.67
9	15652	223.80	431.55	517.86	224.06
10	15653	223.80	372.84	447.40	224.06
...	...	...	...	...	...

为了可视化说明数据分布，本文在 MATLAB 中对数据进行了描绘，代码见附录。所得图

形根据特征称为十二峰分布图。

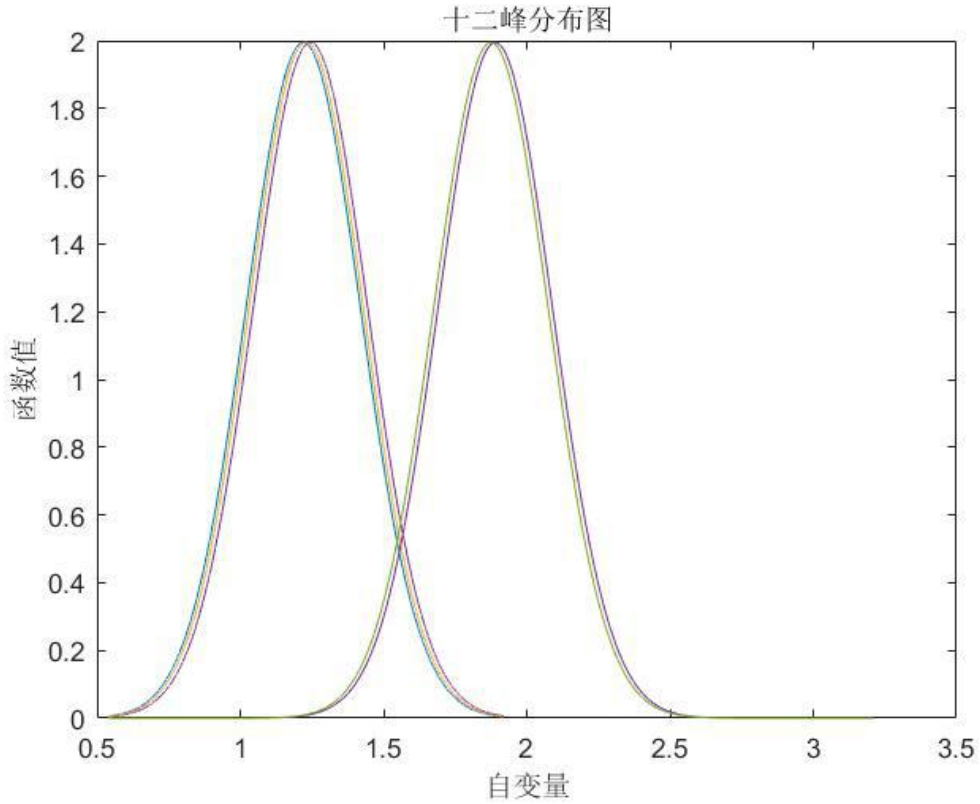

图 5.8 十二峰分布图

从图 5.8 中，我们发现 12 种组合的正态分布整体呈现出双峰趋势线，由此我们也可把它成为双峰分布。

5.3.5 调价后的定价评价

参考问题二中对已经成交货运线路历史交易数据中的定价用数据包络模型进行评价，在问题三中对线路任务三次报价进行评价，本人仍使用数据包络模型来进行评价。在上面的多元线性回归分析中，可知指标需求类型 2 和车辆吨位和线路任务的定价线性无关，因此本文在拟合数据集 2 中剔除这两项指标，并把线路任务三次定价指标的数据完善进去。以总里程、业务类型、需求类型 2、执行时间、车辆长度、打包类型、运输等级、计划卸货等待时长、计划时长、交易对象、需求紧急程度、线路总成本作为投入指标，以第一次报价、第二次报价、第三次报价为产出指标输入问题二中的评价模型，评价结果分数越高说明线路成本越低，报价越高，平台的收益也就越高。

求解模型延用问题二中最小二乘法算法进行求解，求解得到中对线路任务三次报价的评价分数结果分析如下（具体数据见附录）：

任务 id17504 的评价结果分数最低为 0.0125，评价结果分数为 1 的任务 id 有 31 个，

占总体的 2%，这部分任务平台所有任务 id 的定价是最合理的，最有利于平台，平台收益最大。存在 57 个任务 id 评价结果分数大于 0.5，占比 3.8%，这部分定价策略较合理，资源配置利用程度高。

5.4 问题四的分析

尊敬的 A 无车承运人平台：

在 2020 年第十届 MatherCup 高校数学建模挑战赛中，我们团队有幸对贵平台的线路定价问题进行研究。运输平台的竞争是激烈的，如何平衡低成本和订单成交速度是限制平台发展速度与规模的重要议题。我们冒昧的提出了一些关于线路定价及调价优化的建议，不足之处还请批评指正。

一、优化货运线路指导价

通过对历史成交数据采用因子分析法及主成分分析法进行相关性研究发现，贵平台货运线路定价的主要因素集中在线路总成本、总里程、需求类型及业务类型等方面。其中线路成本价与线路定价的相关性显著水平达到 0.987，具有较高的指导意义。因此，我们建议从以上角度出发，综合运用数据挖掘和机器学习等方法尽可能地提高指导价的准确性以降低议价成本。

二、施行差异化线路调价

通过对线路历史交易数据进行详细的分析，我们结合数据包络这一种客观评价模型，辅以多元线性回归考虑所得影响因素的相关性对贵平台的定价及调价做了效果评价，并提出了相应的调价策略，其中有 31 个评价结果为满分 1 的方案使得平台收益最大，值得平台深入借鉴。该模型以面向标签化对象的方式来精准调价，能够提升交易达成的速度，我们希望该模型能够满足贵平台的实际需求，并为定价与调价提供一些参考。

三、推动建立 C 类承运方联盟

在对贵平台的指导报价和成交价进行分析时，我们发现贵平台上 C 类承运方运输的线路价格波动较大，不利于贵平台对于车源的长期维护。因此，我们建议通过联合 C 类承运方的方式实现规模化运营，在缓解恶性竞争和提升报价规范化的同时降低平台运营成本。

以上建议是我们团队的个人想法，希望能对贵平台有所帮助。祝您企业蓬勃发展！

此致

敬谢！

MatherCup 参赛团队

2020 年 5 月 25 日

六、模型评价

6.1 模型优点

1、本文通过构建模型对问题进行分析，考虑问题的角度较为全面，从不同的模型切入，同时进行了模型的检验，建模过程较为严谨。

2、本文采用元线性回归法对各因子进行回归，对变量进行剔除，保证了各变量均为显著（$p<0.05$）即各自变量均对结果具有良好的解释性，模型拟合度极高，且模型整体显著性很好。

3、采用多重插补算法对缺失数据进行填补，降低了缺失数据导致对评价结果造成的误差。

6.2 模型缺点

1、本文为了求解方便，对于一些因素做了假设理想的情况，与实际情况有偏差，结果与实际也有一定的误差。

2、在分析影响定价影响因素时，只分析了最为重要的几个方面，忽略了其他客观因素的影响，过于理想化。

七、模型推广

本文中运用的“数据包络模型”模型，在现实生活中有较广的运用，除了在本文中的“定价评价”问题上，得到了很好的体现以外，在现实生活中的其他领域，运用也较多，比如：“农业数据评价”领域（和“交通数据评价”领域）。

参考文献

[1]韩中庚，数学建模方法及其应用[M]，北京：高等教育出版社，2005。

[2]吴建国、汪名杰、李虎军等，数学建模案例精编[M]，北京：中国水利水电出版社，2005.

[3]贺定修、冯天祥等，数学模型基础[M]，成都：西南交通大学出版社，2011.

[4]王冬琳等，数学建模及实验[M]，北京：国防工业出版社，2005.

[5]司守奎，孙兆亮.数学建模算法与应用（第二版）[M].北京：国防工业出版社，2017.4.

[6]薛薇，SPSS 统计分析方法及应用（第 3 版）[M]，北京，电子工业出版社，2013.1

[7]Liang L,Yang F,Cook W D,Zhu J.DEA models for supply chain efficiency evaluation. Annals of Operations Research, 2006，145(1):35-49.

[8]朱又亮. 铁路客运列车线路使用费定价探讨[J]. 中国总会计师, 2017(6).

[9]秦锋.用于公路桥梁可靠性评估的车辆荷载多峰分布概率模型分析[J].交通世界,2016(23):50-51.

[10]韩志国,陈智高,王基铭.离散样本数据的多峰分布测度及其应用[J].华东理工大学学报(自然科学版),2010,36(01):130-133.

[11]Himadri Ghosh,Prajneshu. Statistical learning theory for fitting multimodal distribution to rainfall data: an application[J]. Journal of Applied Statistics,2011,38(11).

附录

问题二相关代码

```
import xlrd
from xlrd import *
import xlwt
import xlsxwriter
from xlutils.copy import copy
import numpy as np
from scipy.optimize import fmin_slsqp
class CCR():
    def readexcel(self):
        workbook = xlrd.open_workbook(r'C:\Users\Administrator\Desktop\软件实现\123123(1).xls')
        data = workbook.sheet_by_name('data')
        self.D = data.nrows - 1
        L = data.ncols - 1
        self.I = int(input("请输入投入指标的个数:"))
        self.O = L - self.I
        X = []
        Y = []
        for d in range(self.D):
            t = data.row_values(d+1, 1, 9)
            tt = data.row_values(d+1, 9, 11)
            X.append(t)
            Y.append(tt)
        self.X = np.array(X)
        self.Y = np.array(Y)
        self.output_w = np.zeros((self.O,1), dtype=np.float)
        self.input_w = np.zeros((self.I,1), dtype=np.float)
        self.lambdas = np.zeros((self.D,1), dtype=np.float)
        self.efficiency = np.zeros((self.D,1), dtype=np.float)

    def __efficiency(self, unit):
        p = np.dot(self.X,self.input_w)
        q = np.dot(self.Y,self.output_w)

        return  (q/p)[unit]

    def __target(self,x,unit):
        in_w, out_w, lambdas = x[:self.I], x[self.I:(self.I+self.O)], x[(self.I+self.O):]
        p = np.dot(self.X[unit], in_w)
        q = np.dot(self.Y[unit], out_w)
```

```
        return  q/p
    def __constrains(self,x,unit):
        in_w, out_w, lambdas = x[:self.I], x[self.I:(self.I + self.O)],
x[(self.I + self.O):]
        constr = []
        for i in range(self.I):
            t = self.__target(x,unit)
            lhs = np.dot(self.X[:,i],lambdas)
            cons = t*self.X[unit,i] - lhs
            constr.append(cons)

        for o in range(self.O):
            lhs = np.dot(self.Y[:,o],lambdas)
            cons = lhs - self.Y[unit,o]
            constr.append(cons)

        for d in range(self.D):
            constr.append((lambdas[d]))
        return np.array(constr)

    def __optimize(self):

        d0 = self.D + self.I + self.O
        for d in range(self.D):
            x0 = np.random.rand(d0) - 0.5
            x0 =
fmin_slsqp(self.__target,x0,f_ieqcons=self.__constrains,args=(d,))
            self.input_w,self.output_w,self.lambdas =
x0[:self.I],x0[self.I:(self.I+self.O)],x0[(self.I+self.O):]
            self.efficiency[d] = self.__efficiency(d)

    def fit(self):
        self.__optimize()
        return  self.efficiency
if __name__ == '__main__':
    ccr = CCR()
    ccr.readexcel()
    rs = ccr.fit()
    zz = ccr.efficiency
    print(ccr.efficiency)
    workbook = xlsxwriter.Workbook('987.xlsx')
    worksheet = workbook.add_worksheet('sheet1')
    P = len(zz)
    for p in range(P):
```

```
            worksheet.write(p,1,zz[p])
        workbook.close()
```

问题三相关代码

第一组 Python 代码:

```
import numpy as np
import math
import pandas as pd
import random
class Logistic:
    matrix = ()  # 读入数据矩阵
    test_matrix = ()  # 测试数据矩阵
    y = ()  # 分类情况，y[i]表示第 i 组数据的分类情况
    test_y = ()  # 测试数据集的分类情况
    x = ()  # 特征矩阵，其中 x[i]表示第 i 个实例的特征取值情况,最后一维为 1
    test_x = ()  # 测试数据集的特征矩阵
    w = ()  # 对应扩充特征后的 w
    n = 0  # 特征数的个数，其中 w 是 n+1 维的
    dataSum = 0  # 数据量
    testSum = 0  # 测试数据集大小

    # sigmoid 函数
    @staticmethod
    def sig(wx):
        if wx < -10:
            return 0
        else:
            return 1 / (1 + math.exp(-wx))

    # 计算对数似然的值，不加正则，梯度上升法，没有加负号
    def cal_loss1(self):
        loss = 0
        for i in range(self.dataSum):
            w_multi_x = np.dot(self.x[i], self.w)
            # print(w_multi_x)
            loss -= np.dot(self.y[i], w_multi_x)
            # 防止溢出，所以对 wx 进行讨论
            if w_multi_x > 0:
                loss += w_multi_x + math.log(1 + math.exp(-w_multi_x))
            else:
                loss += math.log(1 + math.exp(w_multi_x))
        return loss

    # 计算损失函数的值，加正则，梯度下降法，加负号
    def cal_loss2(self, regex):
```

```
        loss = 0
        for i in range(self.dataSum):
            # print(self.x[i])
            w_multi_x = np.dot(np.mat(self.x[i]), self.w)
            # print(w_multi_x)
            loss -= np.dot(self.y[i], w_multi_x)
            # 防止溢出，所以对 wx 进行讨论
            if w_multi_x > 0:
                loss += w_multi_x + math.log(1 + math.exp(-w_multi_x))
            else:
                loss += math.log(1 + math.exp(w_multi_x))
        loss += regex * np.dot(self.w.T, self.w)[0, 0]
        # loss /= self.dataSum
        return loss


    # 计算梯度 ，随机下降法
    def cal_gradient1(self):
        gradient = np.zeros((self.n + 1, 1))
        i = random.randint(0, self.dataSum - 1)
        wx = np.dot(np.mat(self.x[i]), self.w)
        for j in range(self.n + 1):
            gradient[j][0] += self.x[i][j] * (-self.y[i] + Logistic.sig(wx))
        return gradient

    # 计算梯度，带正则，损失函数的梯度
    def cal_gradient2(self, regex):
        gradient = np.zeros((self.n + 1, 1))
        i = random.randint(0, self.dataSum - 1)
        wx = np.dot(np.mat(self.x[i]), self.w)
        for j in range(self.n + 1):
            gradient[j][0] += self.x[i][j] * (-self.y[i] + Logistic.sig(wx))
        gradient += regex * self.w
        # print(gradient)
        # gradient /= self.dataSum
        # print(gradient)
        return gradient

    # 使用梯度下降法优化参数，似然函数，不带正则
    def de_gradient1(self, lamda, door):
        # print(self.w)
        loss0 = self.cal_loss1()
        g0 = self.cal_gradient1()
        w0 = self.w
```

```
        self.w -= lamda * g0
        loss1 = self.cal_loss1()
        cnt = 0
        while cnt < door:
            cnt += 1
            loss0 = loss1
            g0 = self.cal_gradient1()
            w0 = self.w
            self.w -= lamda * g0
            loss1 = self.cal_loss1()
            # print(loss0 - loss1)
        self.w = w0
        # print(self.w)
        # 返回损失函数的值
        return loss0

    # 使用梯度下降法求解带正则项的 w
    def de_gradient2(self, lamda, door, regex):
        loss0 = self.cal_loss2(regex)
        g0 = self.cal_gradient2(regex)
        w0 = self.w
        self.w -= lamda * g0
        loss1 = self.cal_loss2(regex)
        cnt = 0
        while cnt < door:
            # print(loss1 - loss0)
            # print(g0)
            cnt += 1
            loss0 = loss1
            g0 = self.cal_gradient2(regex)
            w0 = self.w
            self.w -= lamda * g0
            loss1 = self.cal_loss2(regex)
        self.w = w0
        # 返回损失函数的值
        return loss0

    # 读取训练集
    def read_data(self, file):
        self.matrix = pd.read_csv(file, header=1).values
        # print(self.matrix)
        # with open(file) as f:
        #     self.matrix = np.loadtxt(f, float, delimiter=",")
        self.dataSum = len(self.matrix)
```

```
        self.n = len(self.matrix[0]) - 1
        add = np.ones((self.dataSum, 1))
        self.x = np.hstack((self.matrix[:, :self.n], add))
        # print(self.x)
        self.y = self.matrix[:, self.n]
        self.w = np.ones((self.n + 1, 1))

    # 读取测试集
    def read_test_data(self, file):
        self.test_matrix = pd.read_csv(file, header=1).values
        # with open(file) as f:
        #     self.test_matrix = np.loadtxt(f, float, delimiter=',')
        self.testSum = len(self.test_matrix)
        self.test_x = np.hstack((self.test_matrix[:, :self.n],
np.ones((self.testSum, 1))))
        self.test_y = self.test_matrix[:, self.n]

    # 预测
    def pre_test(self):
        cnt = 0
        for i in range(self.testSum):
            pre_wx = np.dot(np.mat(self.test_x[i]), self.w)
            # print(pre_wx)
            if (pre_wx >= 0) and (self.test_y[i] == 1):
                cnt += 1
            elif (pre_wx <= 0) and (self.test_y[i] == 0):
                cnt += 1
        return cnt / self.testSum
def test_model():
    # 测试模型
    test = Logistic()
    train_set = "gauss4.csv"
    test_set = "test_gauss.csv"
    test.read_data(train_set)
    lamda = 1e-2
    steps = 10
    regex = 1e-3
    # test.de_gradient2(lamda, steps, regex)
    test.de_gradient1(lamda, steps)

    test.read_test_data(test_set)
    correct = test.pre_test()
    print(correct)
```

```
    print(test.w)
test_model()
```

第二组 Python 代码：

```
import numpy as np
import math
import pandas as pd
import random
class Logistic:
    matrix = []  # 读入数据矩阵
    test_matrix = []  # 测试数据矩阵
    y = []  # 分类情况，y[i]表示第 i 组数据的分类情况
    test_y = []  # 测试数据集的分类情况
    x = []  # 特征矩阵，其中 x[i]表示第 i 个实例的特征取值情况,最后一维为 1
    test_x = []  # 测试数据集的特征矩阵
    w = []  # 对应扩充特征后的 w
    n = 0  # 特征数的个数，其中 w 是 n+1 维的
    dataSum = 0  # 数据量
    testSum = 0  # 测试数据集大小

    # sigmoid 函数
    @staticmethod
    def sig(wx):
        if wx < -10:
            return 0
        else:
            return 1 / (1 + math.exp(-wx))

    # 计算对数似然的值，不加正则，梯度上升法，没有加负号
    def cal_loss1(self):
        loss = 0
        for i in range(self.dataSum):
            w_multi_x = np.dot(self.x[i], self.w)
            # print(w_multi_x)
            loss -= np.dot(self.y[i], w_multi_x)
            # 防止溢出，所以对 wx 进行讨论
            if w_multi_x > 0:
                loss += w_multi_x + math.log(1 + math.exp(-w_multi_x))
            else:
                loss += math.log(1 + math.exp(w_multi_x))
        return loss

    # 计算损失函数的值，加正则，梯度下降法，加负号
    def cal_loss2(self, regex):
```

```
        loss = 0
        for i in range(self.dataSum):
            # print(self.x[i])
            w_multi_x = np.dot(np.mat(self.x[i]), self.w)
            # print(w_multi_x)
            loss -= np.dot(self.y[i], w_multi_x)
            # 防止溢出，所以对 wx 进行讨论
            if w_multi_x > 0:
                loss += w_multi_x + math.log(1 + math.exp(-w_multi_x))
            else:
                loss += math.log(1 + math.exp(w_multi_x))
        loss += regex * np.dot(self.w.T, self.w)[0, 0]
        # loss /= self.dataSum
        return loss

    # 计算梯度 ，随机下降法
    def cal_gradient1(self):
        gradient = np.zeros((self.n + 1, 1))
        i = random.randint(0, self.dataSum - 1)
        wx = np.dot(np.mat(self.x[i]), self.w)
        for j in range(self.n + 1):
            gradient[j][0] += self.x[i][j] * (-self.y[i] + Logistic.sig(wx))
        return gradient

    # 计算梯度，带正则，损失函数的梯度
    def cal_gradient2(self, regex):
        gradient = np.zeros((self.n + 1, 1))
        i = random.randint(0, self.dataSum - 1)
        wx = np.dot(np.mat(self.x[i]), self.w)
        for j in range(self.n + 1):
            gradient[j][0] += self.x[i][j] * (-self.y[i] + Logistic.sig(wx))
        gradient += regex * self.w
        # print(gradient)
        # gradient /= self.dataSum
        # print(gradient)
        return gradient

    # 使用梯度下降法优化参数，似然函数，不带正则
    def de_gradient1(self, lamda, door):
        # print(self.w)
        loss0 = self.cal_loss1()
        g0 = self.cal_gradient1()
        w0 = self.w
        self.w -= lamda * g0
```

```
    loss1 = self.cal_loss1()
    cnt = 0
    while cnt < door:
        cnt += 1
        loss0 = loss1
        g0 = self.cal_gradient1()
        w0 = self.w
        self.w -= lamda * g0
        loss1 = self.cal_loss1()
        # print(loss0 - loss1)
    self.w = w0
    # print(self.w)
    # 返回损失函数的值
    return loss0

# 使用梯度下降法求解带正则项的 w
def de_gradient2(self, lamda, door, regex):
    loss0 = self.cal_loss2(regex)
    g0 = self.cal_gradient2(regex)
    w0 = self.w
    self.w -= lamda * g0
    loss1 = self.cal_loss2(regex)
    cnt = 0
    while cnt < door:
        # print(loss1 - loss0)
        # print(g0)
        cnt += 1
        loss0 = loss1
        g0 = self.cal_gradient2(regex)
        w0 = self.w
        self.w -= lamda * g0
        loss1 = self.cal_loss2(regex)
    self.w = w0
    # 返回损失函数的值
    return loss0

# 读取训练集
def read_data(self, file):
    self.matrix = pd.read_csv(file, header=1).values
    # print(self.matrix)
    # with open(file) as f:
    #     self.matrix = np.loadtxt(f, float, delimiter=",")
    self.dataSum = len(self.matrix)
    self.n = len(self.matrix[0]) - 1
```

```
        add = np.ones((self.dataSum, 1))
        self.x = np.hstack((self.matrix[:, :self.n], add))
        # print(self.x)
        self.y = self.matrix[:, self.n]
        self.w = np.ones((self.n + 1, 1))

    # 读取测试集
    def read_test_data(self, file):
        self.test_matrix = pd.read_csv(file, header=1).values
        # with open(file) as f:
        #     self.test_matrix = np.loadtxt(f, float, delimiter=',')
        self.testSum = len(self.test_matrix)
        self.test_x = np.hstack((self.test_matrix[:, :self.n],
np.ones((self.testSum, 1))))
        self.test_y = self.test_matrix[:, self.n]

    # 预测
    def pre_test(self):
        cnt = 0
        for i in range(self.testSum):
            pre_wx = np.dot(np.mat(self.test_x[i]), self.w)
            # print(pre_wx)
            if (pre_wx >= 0) and (self.test_y[i] == 1):
                cnt += 1
            elif (pre_wx <= 0) and (self.test_y[i] == 0):
                cnt += 1
        return cnt / self.testSum


def test_model():
    # 测试模型
    test = Logistic()
    train_set = "gauss5.csv"
    test_set = "test_gauss.csv"
    test.read_data(train_set)
    lamda = 1e-2
    steps = 10
    regex = 1e-3
    # test.de_gradient2(lamda, steps, regex)
    test.de_gradient1(lamda, steps)

    test.read_test_data(test_set)
    correct = test.pre_test()
    print(correct)
```

```
    print(test.w)

def generate_data():
    # 生成高斯数据
    f = open('test_gauss_not_bayes.csv', 'w')
    mean0 = [2, 3]
    cov = np.mat([[2, 1], [1, 2]])
    x0 = np.random.multivariate_normal(mean0, cov, 500).T

    mean1 = [7, 8]
    x1 = np.random.multivariate_normal(mean1, cov, 500).T

    for i in range(len(x0.T)):
        line = []
        line.append(x0[0][i])
        line.append(x0[1][i])
        line.append(1)
        line = ",".join(str(i) for i in line)
        line = line + "\n"
        f.write(line)

    for i in range(len(x0.T)):
        line = []
        line.append(x1[0][i])
        line.append(x1[1][i])
        line.append(0)
        line = ",".join(str(i) for i in line)
        line += "\n"
        f.write(line)
    f.close()

test_model()
```

调价策略可视化 MATLAB 代码：

```
clc;
clear;
x1 = 0.54:0.0001:1.90
y1 = normpdf(x1,1.22,0.2)
h1=plot(x1,y1)
hold on
x2 = 0.57:0.0001:3.21
y2 = normpdf(x2,1.89,0.2)
h2=plot(x2,y2)
hold on
```

```
x3 = 0.54:0.0001:3.21
y3 = normpdf(x3,1.875,0.2)
h3=plot(x3,y3)
hold on
x4 = 0.54:0.0001:1.92
y4 = normpdf(x4,1.245,0.2)
h4=plot(x4,y4)
hold on
x5 = 0.57:0.0001:3.21
y5 = normpdf(x5,1.89,0.2)
h5=plot(x5,y5)
hold on
x6 = 0.54:0.0001:3.21
y6 = normpdf(x6,1.875,0.2)
h6=plot(x6,y6)
hold on
x7 = 0.54:0.0001:1.92
y7 = normpdf(x7,1.23,0.2)
h7=plot(x7,y7)
hold on
x8 = 0.57:0.0001:3.21
y8 = normpdf(x8,1.89,0.2)
h8=plot(x8,y8)
hold on
x9 = 0.54:0.0001:3.21
y9 = normpdf(x9,1.875,0.2)
h9=plot(x9,y9)
hold on
x10 = 0.54:0.0001:1.92
y10 = normpdf(x10,1.23,0.2)
h10=plot(x10,y10)
hold on
x11 = 0.57:0.0001:3.21
y11 = normpdf(x11,1.89,0.2)
h11=plot(x11,y11)
hold on
x12 = 0.54:0.0001:3.21
y12 = normpdf(x12,1.875,0.2)
h12=plot(x12,y12)
title('十二峰分布图')
xlabel('自变量')
ylabel('函数值')
```

模型评价 Python 代码

```
import xlrd
from xlrd import *
import xlwt
import xlsxwriter
from xlutils.copy import copy
import numpy as np
from scipy.optimize import fmin_slsqp
class CCR():
    def readexcel(self):
        workbook = xlrd.open_workbook(r'C:\Users\Administrator\Desktop\软件实现\123123(1).xls')
        data = workbook.sheet_by_name('data')
        self.D = data.nrows - 1
        L = data.ncols - 1
        self.I = int(input("请输入投入指标的个数:"))
        self.O = L - self.I
        X = []
        Y = []
        for d in range(self.D):
            t = data.row_values(d+1, 1, 13)
            tt = data.row_values(d+1, 13, 16)
            X.append(t)
            Y.append(tt)
        self.X = np.array(X)
        self.Y = np.array(Y)
        self.output_w = np.zeros((self.O,1), dtype=np.float)
        self.input_w = np.zeros((self.I,1), dtype=np.float)
        self.lambdas = np.zeros((self.D,1), dtype=np.float)
        self.efficiency = np.zeros((self.D,1), dtype=np.float)


    def __efficiency(self, unit):
        p = np.dot(self.X,self.input_w)
        q = np.dot(self.Y,self.output_w)

        return  (q/p)[unit]

    def __target(self,x,unit):
        in_w, out_w, lambdas = x[:self.I], x[self.I:(self.I+self.O)], x[(self.I+self.O):]
        p = np.dot(self.X[unit], in_w)
        q = np.dot(self.Y[unit], out_w)

        return  q/p
    def __constrains(self,x,unit):
```

```
            in_w, out_w, lambdas = x[:self.I], x[self.I:(self.I + self.O)], x[(self.I + self.O):]
            constr = []
            for i in range(self.I):
                t = self.__target(x,unit)
                lhs = np.dot(self.X[:,i],lambdas)
                cons = t*self.X[unit,i] - lhs
                constr.append(cons)

            for o in range(self.O):
                lhs = np.dot(self.Y[:,o],lambdas)
                cons = lhs - self.Y[unit,o]
                constr.append(cons)

            for d in range(self.D):
                constr.append((lambdas[d]))

            return np.array(constr)

        def __optimize(self):

            d0 = self.D + self.I + self.O
            for d in range(self.D):
                x0 = np.random.rand(d0) - 0.5
                x0 = fmin_slsqp(self.__target,x0,f_ieqcons=self.__constrains,args=(d,))
                self.input_w,self.output_w,self.lambdas                                    =
x0[:self.I],x0[self.I:(self.I+self.O)],x0[(self.I+self.O):]
                self.efficiency[d] = self.__efficiency(d)

        def fit(self):
            self.__optimize()
            return    self.efficiency
if __name__ == '__main__':
        ccr = CCR()
        ccr.readexcel()
        rs = ccr.fit()
        zz = ccr.efficiency
        print(ccr.efficiency)
        workbook = xlsxwriter.Workbook('321.xlsx')
        worksheet = workbook.add_worksheet('sheet1')
        P = len(zz)
        for p in range(P):
            worksheet.write(p,1,zz[p])
        workbook.close()
```