队伍编号	204051
题号	A

无车承运人平台线路定价问题

摘 要

无车承运人是基于“互联网+物流”发展起来的一种货运市场运作模式，运输效率高、运营成本低、智能化水平高。但由于目前无车承运人模式的定价机制不够清晰，这将对该模式的进一步发展造成一定的阻碍。因此，本文主要采用机器学习、贝叶斯理论和对抗训练方法建立模型对无车承运人平台线路定价进行深入研究，主要解决了如下若干问题：

问题 1，利用机器学习模型，确定无车承运人平台线路定价的主要因素。本文首先进行了缺失值处理和简单的特征工程，然后基于多因素回归分析、LASSO、随机森林、GBDT 和 XGBoost 建立回归模型并得到各变量重要度。最终确定了线路定价的主要因素有：1 级运输，始发地为陕西省，车辆长度，目的地为新疆省，总里程，车辆吨位，地区 4，从发车到到达的时间，分拨月份和标的月份。

问题 2，通过对比分析附件 1 中的线路指导价（定价）和线路价格（成交价），本文假定当成交价在定价的 95%到 105%之间则认为价格一致。我们按照调价比例，将附件 1 样本分为 3 类：定价较低、定价较高和定价适中，然后基于逻辑回归和 XGBoost 建立分类模型并筛选重要变量，分别利用贝叶斯后验概率和 Kruskal-Wallis 检验对定价较低或较高的原因进行了分析。

问题 3，本文给出的第一次报价为了降低平台运营成本，第二次和第三报价促使快速交易和提高成交率。对于对第一次报价，我们建立以线路指导价为标签的回归模型，给出尽可能接近平台理想的报价；第二次报价我们以成交价为标签，挖掘建模特征与成交价之间的关系，更加精准的给出能快速成交的价格；第三次报价我们采用对抗训练的思想，对成交价随机添加扰动，增强模型泛化能力，给出第三次报价。基于我们的动态定价策略，第一次报价为了降低承运成本仅有 21%的成交率，综合第二三次给出的报价能使最终成交率达到 96%以上。

问题 4，根据研究结果，由于地区的差异化会导致交通运输量的不同，因此可以结合总里程和始发地目的地两个因素，对始发地和目的地较偏远的地方由于交通不便利，可以适当提高线路价格；也可以从总里程的角度出发，结合地区交通因素制定每公里具体的线路价格。

关键词：机器学习，对抗训练，定价策略，变量重要度，贝叶斯后验概率

目 录

1 问题重述

1.1 研究背景

无车承运人平台是基于“互联网+物流”逐步兴起的一种智能化货运作模式，其借助先进数据处理技术将货源、车源及相关信息高效链接，降低运营成本，提高运输效率。但由于该模式目前仅仅处于试点推行阶段，科学的定价机制是其一个亟待解决的问题。

图 1-1 展示了国内无车承运人的主要运营模式，该模式下有三个主要的参与角色，分为为货主、无车承运人平台以及承运人。作为无车承运人平台，既需要面向货主的运输任务报价，同时也需要面向承运司机进行报价。

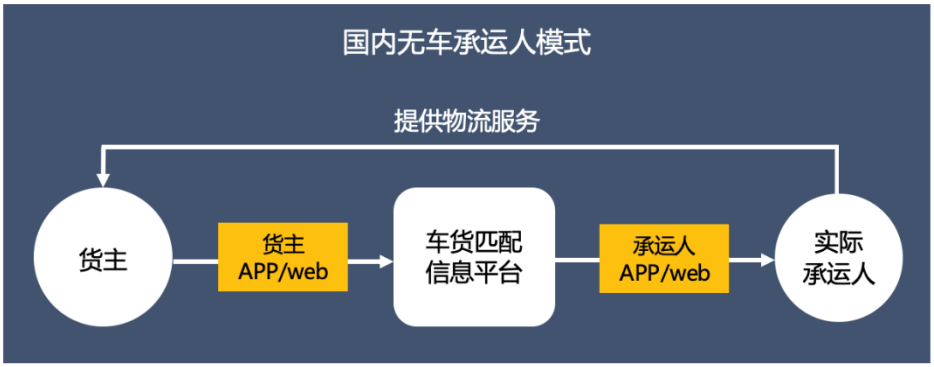

图 1-1 国内无车承运人模式

本研究以无车承运人的视角，暂不考虑面向货主的运输任务的报价，仅面向广大拥有运力资源（货车）的承运司机，将需要承运的线路任务以一定价格提前发布到网络平台上供承运司机浏览并决定是否承运该运输任务。平台采用动态定价的形式保证每个任务必须被承运，若任务未被承运将带来一定损失。作为承运端的司机，会根据平台发布的线路任务和价格进行判断是否接单，司机接单则视为该线路任务交易成功，此线路任务随即从平台下架。若在给定的时间内，该任务没有司机接单，则该线路就可以进行调价。每条线路任务最多允许发布 3 次价格，即首次发布线路价格后仍可刷新两次线路价格，其中附件 1 数据文件中的线路指导价为平台首次发布的线路价格。假设上述线路任务全部为固定车型的整车任务，即一个任务需要由某种车型的 1 辆车完成，不考虑拼载任务。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。

1.2 研究问题

本文需要基于以上背景，根据附件给出的数据（可不限于此），通过数学建模的方法帮助某无车承运人平台解决以下问题：

问题 1：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 2：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 3：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 4：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。

2 符号说明与模型假设

2.1 符号说明

本文使用的符号及其物理意义说明如表 2-1 所示。

表 2-1 符号说明

符号	物理意义
D	数据集
$\boldsymbol{x}_i$	数据集D中的第i个样本
m	数据集样本的个数
N	每个样本$\boldsymbol{x}_i$的属性个数
I_j^{model}	在某个模型（model）下的第j个变量的重要度
$RMSE^{model}$	在某个模型（model）下的均方根误差
I_j	第j个因素的最终的重要度
D_1	随机分层采样所构造的样本集合
C_1	样本类别：线路指导价合适
C_2	样本类别：线路指导价较低
C_3	样本类别：线路指导价较高
a_j	基于分类模型筛选的 10 个变量之一
s	真实成交价
s_1	基于定价策略给出的报价
$deal$	0,1 变量。$deal$等于 0 代表成交，$deal$等于 1 代表成交

2.2 模型假设

为了便于模型构建与求解，在整个解题过程中，假设：

(1) 一个任务需要由某种车型的 1 辆车完成，不考虑拼载任务；

(2) 假设无车承运人平台不考虑面向货主的报价，因此本题仅考虑无车承运人平台向承运司机的报价；

(3) 假设该无车承运人不存在竞争关系；

(4) 假设供需平衡；

(5) 预测成交价在真实成交价的 95%到 105%之间时，则认为可成交;

3 问题一：模型建立与求解

3.1 问题分析与思路

从问题描述中可知，可以采用题目给定的附件 1 中的货运线路运作基本信息为参考提取影响该平台定价的各类别变量。对提取的类别变量，应通过定量分析的方法研究影响无车承运人平台货运线路定价的主要因素有哪些。传统的确定影响因变量的主要因素的方法有 Pearson 相关系数法、方差分析、卡方检验、多因素回归分析等方法。

在机器学习中，确定影响影响因变量的主要因素实际上是一个特征选择问题，而机器学习的特征选择方法除了上述传统方法外，还可以基于预测模型如 LASSO 回归、决策树等机器学习模型进行嵌入式的特征选择。在实践中，由传统的方法筛选变量建立模型并进行预测，结果并不理想，为了提高预测精度，工程师经常采用基于预测模型的特征选择方法。

因此，本文首先利用多因素回归分析给出各个变量的重要度并进行简单的分析。然后基于 LASSO 回归、随机森林（Random Forest）、GBDT（Gradient Boosting Decison Tree，梯度提升树）、XGBoost（eXtreme Gradient Boosting）四种模型分别给出各自模型下的变量重要度，最后通过综合四个方法的变量重要度，最终确定了影响影响无车承运人平台进行货运线路定价的主要因素。问题 1 的求解思路如图 3-1。

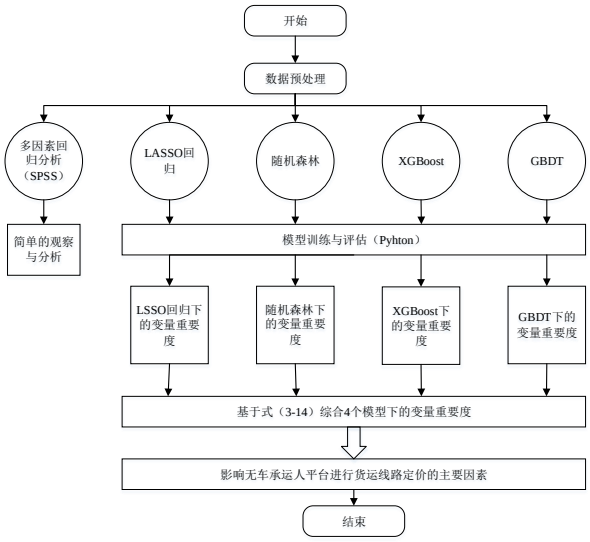

图 3-1 问题 1 求解思路

3.2 模型建立

3.2.1 多因素回归模型

记货运线路的历史交易数据为$\mathrm{D}=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$，其中$\boldsymbol{x}_i=$

$(x_{i1}; x_{i2}; \ldots; x_{iN})$，$m$为数据集中样本的个数，$N$为每个样本的属性的个数。

多因素回归分析是假设自变量与因变量之间存在线性关系，并利用样本数据求得线性表达式的统计分析方法[1]。其一般形式如下：

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i + b \tag{3-1}$$

式中$\boldsymbol{w}^T = (w_1, w_2 \ldots, w_N)$，$b$为常数项。标准化的参数$w_j$大小代表了第$j$个自变量对因变量的影响程度。通过最小化均方误差，即最小化

$$\arg\min_{(\boldsymbol{w},b)} \sum_{i=1}^{m} (f(\boldsymbol{x}_i) - y_i)^2 \tag{3-2}$$

我们就可以确定线性模型的参数。利用归一化公式（3-3）我们就能得到每个变量在多因素回归分析下的重要度如下式。

$$I_j^{model} = \frac{|w_j|}{|w_1|+|w_2|+\cdots+|w_N|} \tag{3-3}$$

3.2.2 嵌入式特征选择

(1) LASSO 回归

为了降低线性回归模型过拟合的风险，通过向式（3-2）引入L_1范数正则化项，得到 LASSO 回归[2]。

$$\arg\min_{(\boldsymbol{w},b)} \sum_{i=1}^{m} (f(\boldsymbol{x}_i) - y_i)^2 + \lambda \|\boldsymbol{w}\|_1 \tag{3-4}$$

式中$\lambda > 0$为正则化参数。通过L_1范数我们会较易得到$\boldsymbol{w}$的“稀疏”解，即$\boldsymbol{w}$的部分分量为 0。

(2) 随机森林

决策树是机器学习中最基础且应用最广泛的算法模型之一，它易于理解、可解释性强，包括了 ID3、C4.5、CART。通过利用集成学习结合多个学习器或模型，可获得比单一学习器显著优越的泛化性能。随机森林就是在集成学习 Bagging 上的扩展变体，使用 CART 作为基学习器，在每个决策树的训练过程中引入了随机属性选择。

对于普通的决策树，我们会在节点上所有的N个样本特征中选择一个最优的特征来做决策树的左右子树划分，但是随机森林通过随机选择节点上的一部分样本特征，这个数字小于N，假设为N_{sub}，然后在这些随机选择的N_{sub}个样本特征中，选择一个最优的特征来做决策树的左右子树划分[3]。这样进一步增强了模型的泛化能力。在回归模型中 CART 采用式（3-5）进行划分属性的选择[4]。

$$\min_{A,S}[\min_{c_1} \sum_{x_i \in D_1(A,S)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in D_2(A,S)} (y_i - c_2)^2] \tag{3-5}$$

式中，A为决策树结点分裂时任意可选的划分属性，s为以A为划分属性的划分点，s将数据集划分为D_1和$D_2$2 个数据集，c_1和c_2分别为D_1和$D_2$2 个数据集的样本输出均值。而**特征重要度就是基于式（3-5）的分裂增益进行计算的**，这样我们就得

到了决策树模型下的变量重要度，包括**随机森林、GBDT、XGBoost**。

(3) GBDT

GBDT（梯度提升树）是在 Boosting 集成策略下的以 CART 回归树为基学习器的决策树算法，使用了前向分步算法进行迭代。在 GBDT 的迭代中，假设我们前一轮得到的强学习器是$f_{t-1}(x)$，损失函数是$L(y, f_{t-1}(x))$，我们本轮迭代的目标是找到一个 CART 回归树模型的弱学习器$h_t(x)$，让本轮的损失函数最小。

$$L\big(y, f_t(x)\big) = L\big(y, f_{t-1}(x) + h_t(x)\big) \tag{3-6}$$

也就是说，本轮迭代找到决策树，要让样本的损失量尽量变小。对于损失函数的拟合问题，Freidman[5]提出了用损失函数的负梯度来拟合本轮损失的近似值，进而拟合一个 CART 回归树。第t轮的第i个样本的损失函数的负梯度表示为

$$r_{ti} = -\left[\frac{\partial L(y_i, f_t(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)} \tag{3-7}$$

利用$(x_i, r_{ti}), (i = 1,2,..m)$，我们可以拟合一颗 CART 回归树，得到了第$t$颗回归树，其对应的叶结点区域 $(R_{tj}, j = 1,2,..J)$。其中J为叶子节点的个数。

针对每一个叶子节点里的样本，我们求出使损失函数最小，也就是拟合叶子节点最好的输出值c_{tj}：

$$c_{tj} = \arg\min_{c} \sum_{x_i \in R_{tj}} L(y_i, f_{t-1}(x_i) + c) \tag{3-8}$$

这样我们就得到了本轮的决策树拟合函数如下：

$$h_t(x) = \sum_{j=1}^{J} c_{tj} I(x \in R_{tj}) \tag{3-9}$$

从而本轮最终得到的强学习器为：

$$f_t(x) = f_{t-1}(x) + \sum_{j=1}^{J} c_{tj} I(x \in R_{tj}) \tag{3-10}$$

(4) XGBoost

XGBoost 是大规模并行 boosting tree 的工具，是 GBDT 的高效实现。对比原算法 GBDT，XGBoost 主要从下面三个方面做了优化：

(a) 算法本身的优化。在算法的损失函数上，除了本身的损失，还加上了正则化部分。在算法的优化方式上，GBDT 的损失函数只对误差部分做负梯度（一阶泰勒）展开，而 XGBoost 损失函数对误差部分做二阶泰勒展开，更加准确；

(b) 算法运行效率的优化；

(c) 算法健壮性的优化。

在 GBDT 损失函数$L\big(y, f_{t-1}(x) + h_t(x)\big)$的基础上，加入正则化项：

$$\Omega(h_t) = \gamma J + \frac{\gamma}{2}\sum_{j=1}^{J} c_{tj}^2 \tag{3-11}$$

其中c_{tj}与 GBDT 的含义一致。最终 XGBoost 的损失函数可以表达为：

$$L_t = \sum_{i=1}^{m} L\left(y_i, f_{t-1}(x_i) + h_t(x_i)\right) + \gamma J + \frac{\gamma}{2}\sum_{j=1}^{J} c_{tj}^2 \tag{3-12}$$

关于 XGBoost 的优化求解本文具体内容可见参考文献[6]。

3.2.3 重要度的综合

对于以上介绍的 4 种模型，每一种方法都能计算出变量的重要度I_j^{model}，本文根据这 4 种模型的预测性能，即均方根误差：

$$RMSE^{model} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(f(\boldsymbol{x}_i) - y_i)^2} \tag{3-13}$$

对模型下变量的重要度加权求和以得到最终的变量重要度，计算公式如下：

$$I_j = \sum_{model=1}^{4} \frac{1/RMSE^{model}}{\sum_{model=1}^{4} 1/RMSE^{model}} \times I_j^{model} \tag{3-14}$$

式中：I_j为第j个变量最终的重要度。

$$model = \{1: LASSO, 2: \text{随机森林}, 3: GBDT, 4: XGBoost\}$$

3.3 模型求解

3.3.1 数据预处理

(1) 对数据的编码方案见附录 B。

(2) 匹配附件 1 与附件 2 中同时出现的变量（列），去除附件 1 中未在附件 2 中出现的变量（列）

(3) 缺失值处理。去除缺失值超过 90%以上的变量，包括子包号，装卸的次数，装的次数，卸的次数，是否需要装卸。对其余存在缺失值的变量，使用该列的众数替换缺失值，具体包括交易模式分拨时间，始发地省份名称，打包类型，是否续签，目的地省份名称，线路总成本，计划卸货完成时间和计划卸货等待时长。

(4) 对类别变量（除作业路线外）均采取 onehot 处理。对于计划卸货完成时间，计划到达时间，计划发车时间，计划靠车时间，分拨时间，标的_创建日期等时间信息，分别计算相邻两者之间的间隔时长以得到时间段信息，分别提取月份、天、和时刻以得到时间点信息。由于计划卸货完成时间，计划到达时间，计划发车时间，计划靠车时间，分拨时间，标的_创建日期'，标的_创建时间和计划发车日期中的日期格式无法用于建模，提取上述时间段和时间点信息后，去除这些列。

3.3.2 重要度计算

(1) 利用 SPSS 进行多因素回归分析得到各个变量的标准化系数，根据归一化公式（3-3）给出各个变量的重要度如表 3-1 和重要度排序如图 3-2 所示。

表 3-1 多因素回归模型下各变量的重要度

变量	重要度	变量	重要度
总里程	0.438	计划到达的天数	0.017
始发网点	0.106	发车_天	0.016
运输等级	0.103	靠车_分拨_时间	0.010
目的地省份名称	0.052	靠车_小时	0.010
计划卸货的月份	0.035	到达_发车_时间	0.010
地区	0.035	业务类型	0.009
标的创建日期	0.030	发车_小时	0.008
计划发车月份	0.026	发车_靠车_时间	0.008
分拨月份	0.023	卸货_天	0.008
地区类型	0.018	卸货_到达_时间	0.008

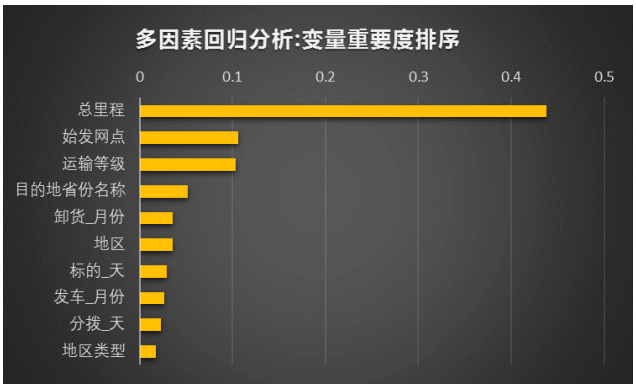

图 3-2 多因素回归模型下各变量的重要度排序

(2) LASSO 回归、随机森林、XGBoost、GBDT 求解

利用 Python 编程分别建立 LASSO 回归、随机森林、XGBoost、GBDT 预测模型，利用附件一基于 5 折交叉验证和网格搜索，分别优化 LASSO 回归、随机森林、XGBoost、GBDT 的超参数，然后基于参数优化后的模型，拟合全部数据，四者的 RMSE 分别为 391.0，47.7，44.6，42.3，同时我们也得到了四个模型下的变量重要度。

我们选取了四个模型下的前 10 个变量的重要度排序，如图 3-3 所示。可以得知简单的线型模型表现较差，而集成算法表现均较为接近。对单个模型变量重要度进行分析时，考虑到 LASSO 准确性较差，因此只考虑后三个模型提供的变量重要度。可以发现，在三个模型中，车辆吨位、车辆长度、车辆的初始地和目的地、总里程、车辆是否从地区 4 出发、从发车到到达的时长以及第几个月分拨，这些因素对定价有较大影响。

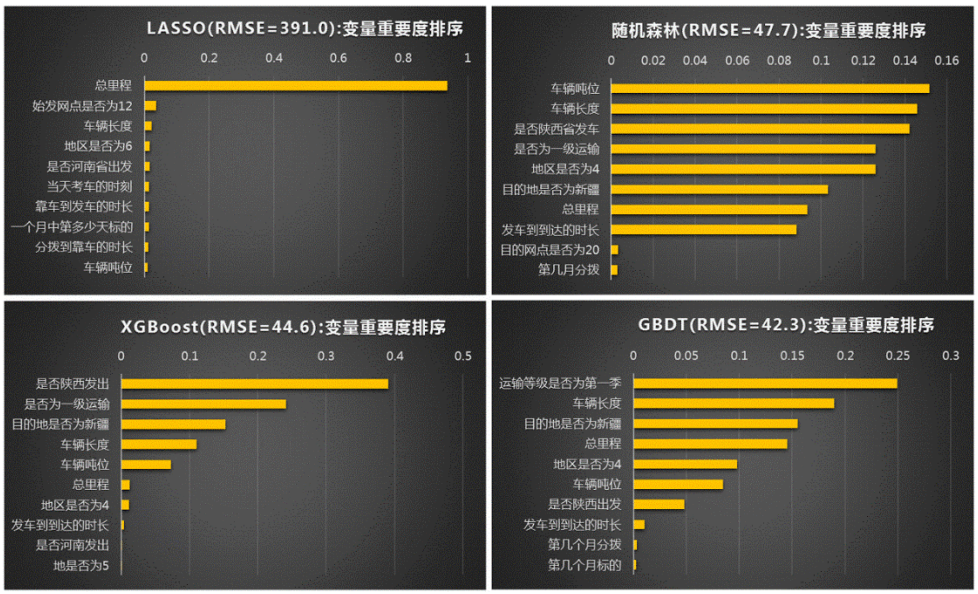

图 3-3 四种模型下的各变量重要度排序

按均方根误差对四个机器学习模型的重要度进行加权，得到重要度前十的变量降序排列为是否为 1 级运输，始发地是否为陕西省，车辆长度，目的地省份是否为新疆省，总里程，车辆吨位，地区是否为 4，车辆行驶时间（从发车到到达的时间），分拨月份和标的月份。总的来看，运输等级，始发地与目的地，车辆大小和载重，路线长度和行车时间，任务发布时间等几个因素对定价影响较大。

4 问题二：模型建立与求解

4.1 问题分析与思路

根据附件 1 给出的“调价类型”信息：成交价在指导价的 95%-105%内都属于未调整。本文按调价比例将样本分成三类：线路指导价合适、线路指导价较低（调价比例大于 1.05）、线路指导价较高（调价比例小于 0.95）。

我们采用了 2 种方法对线路指导价不合适（线路指导价较低或较高）的原因进行了分析。首先我们分别利用逻辑回归（Logistics Regression, LR）和 XGBoost 建立分类预测模型，得出 2 个模型下重要度排序前 10 的变量。

第一种方法，基于后验概率的分析。分别以 LR 和 XGBoost 筛选的 10 个变量为条件变量，计算线路指导价较低和较高的后验概率，从而表征每个筛选出的重要因素的特定取值对线路指导价的影响。

第二种方法，基于 Kruskal-Wallis 检验的分析。首先我们对附件一和附件二的数据进行采样，构造新的样本。具体采样方式为，分别对每个变量进行随机分层采样，构造样本$D_1 = \{(x_{i1}, x_{i2}, \dots, x_{i0}, y_i)\}_{i=1}^{1000}$，重复一千次，得到一千个样本。该一千个样本视为对 10 个主要变量取值的随机组合。这部分样本是对给定变量的取值的组合，因此其中包含部分附件 1 中未出现的新样本。基于附件 1 建立多因素训练分类模型，预测该 1000 个样本分别对应的价格类别。基于 1000 个样本的预测标签，即线路指导价合适、线路指导价较低、线路指导价较高这三个标签，通过 Kruskal-Wallis 检验判断在三种类别下分布显著不同的因素，进而分析影响线路指导价较高和较低的原因。

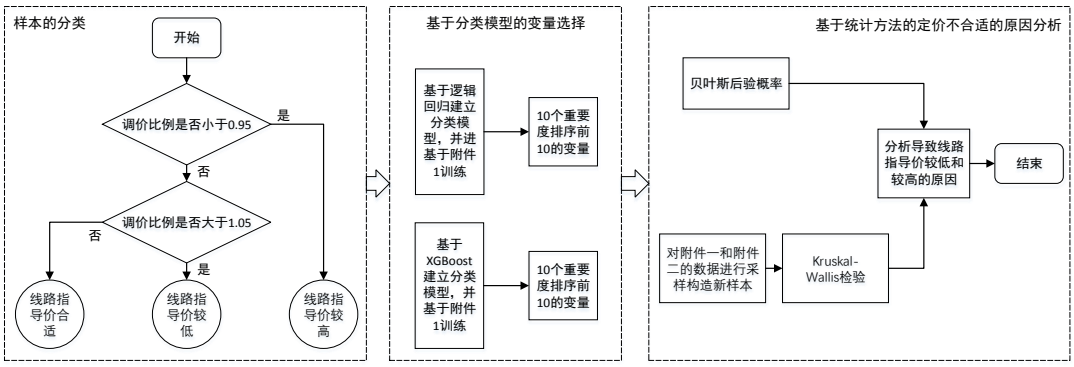

图 4-1 问题 2 求解思路

4.2 模型建立

4.2.1 线路指导价较高和较低的研究方法

（1） 基于后验概率的分析方法

首先，对于附件 1 的数据，本文分成 3 类：C_1, C_2, C_3。C_1对应着平台给出的

第一次指导价与成交价相同，即一次成交；C_2对应着成交价高于第一次指导价，即价格被调高，交易可能在第二次或第三次成交；C_3对应着成交价低于第一次指导价，即价格被调低，同样地，交易可能在第二次或第三次成交。

其次，本文利用逻辑回归和 XGBoost 分别建立**分类模型**，基于附件 1 使用 5 折交叉验证训练模型并优化模型的超参数，并拟合全部数据，通过 F1 值来评价模型的预测精度。

$$F1 = \frac{2 \times P \times R}{P+R} \tag{4-1}$$

式中：P为查准率（precision），R为查全率（recall）。

最后，与上文类似，我们可以根据 2 个预测模型分别得到变量的重要度。设基于逻辑回归或 XGBoost 分类模型筛选出重要度排序前 10 的变量为：$a_1, a_2, \dots, a_{10}$，本文利用式（4-2）计算已知属性变量下价格类别的后验概率，进而可以分析不同取值对价状态的影响，发现导致第一次指导价不能交易成功的原因，或者说什么因素容易使得价格被调高或调低。

$$Pr\{C_k | a_j\} \tag{4-2}$$

式中：$C_k \in \{C_1, C_2, C_3\}$，$a_j \in \{a_1, a_2, \dots, a_{10}\}$。

（2）基于 Kruskal-Wallis 检验的分析方法

多独立样本 Kruskal-Wallis 检验的实质上是两独立样本时的 Mann-Whitney U 检验在多个独立样本下的推广，用于检验多个总体的分布是否存在显著差异。 其原假设是： 多个独立样本来自的多个总体的分布无显著差异。多独立样本 Kruskal-Wallis 检验的基本思想是：首先，将多组样本数混合并按升序排序，求出各变量值的秩；然后，考察各组秩的均值是否存在显著差异。如果各组秩的均值不存在显著差异，则认为多组数据充分混合，数值相差不大，可以认为多个总体的分布无显著差异；反之，如果各组秩的均值存在显著差异，则是多组数据无法混合，有些组的数值普遍偏大，有些组的数值普遍偏小，可认为多个总体的分布存在显著差异，至少有一个样本不同于其他样本。Kruskal-Wallis 检验不仅可以检验连续变量，对于离散变量也同样适用。

因此本文可利用分析 Kruskal-Wallis 检验，判断逻辑回归和 XGBoost 筛选出的变量在三种类别下（C_1, C_2, C_3）是否具有显著性差异，进而对有显著性的变量进行分析，得出线路指导价不合适的原因。

4.3 已成交交易定价评价

附件一的样本在三种类别（C_1, C_2, C_3）下的占比如图 4-2 所示。

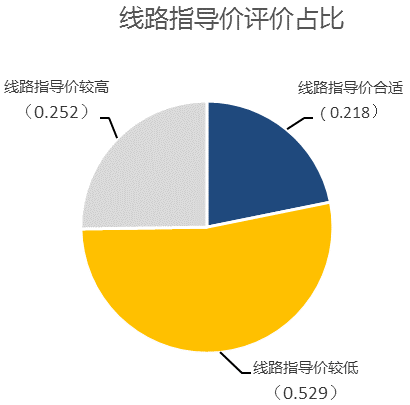

图 4-2 线路指导价评价结果占比

由图 4-2 可知，对比分析附件一中的线路指导价（定价）和线路价格（成交价），可以发现仅 21.8%的任务的成交价与定价一致，25.2%的任务的成交价低于定价，52.9%的任务的成交价高于定价。这说明平台所发布的定价往往偏低，是是为了节省成本，增加平台获益。

4.4 线路指导价较低和较高的原因

4.4.1 基于后验概率的分析结果

基于附件 1 进行 5 折交叉验证，我们得到逻辑回归和 XGBoost 模型的 f1 值分别为 0.85 和 0.985，这说明我们的模型具有较高的准确性，从而为本文的后续分析奠定了基础。

基于逻辑回归筛选出的十个变量分别为：为从发车至到达的小时数，计划发车发车月份，计划到达时刻，地区是否为 6，计划卸货时刻，线路编码，分拨月份，标的月份，计划靠车月份，计划靠车时刻；XGBoost 筛选出的十个变量为目的网点是否为 17，车辆吨位，目的网点是否为 16，始发网点是否为 18，车辆长度，地区是否为 2，地区是否为 5，总里程，目的地是否为河南和始发网点是否为 6。我们可以发现通过逻辑回归筛选的变量主要是时间因素，而 XGBoost 筛选出的变量主要为地区因素。

通过式（4-2）分别计算 2 个模型下价格状态类别的后验概率，我们得到了各个特征作为条件变量在什么取值下使得后验概率最大，表 4-1 对应着线路指导价较低的后验概率$Pr\{C_2|a_j\}$，表 4-2 对应线路指导价较高的后验概率$Pr\{C_3|a_j\}$。

表 4-1 两个模型下的各个变量最大后验概率对应取值

定价较低（成交价/定价>1.05）			
逻辑回归		**XGBoost**	
变量	最大后验概率及其对应取值	变量	最大后验概率及其对应取值
从发车到到	不少于一个小时（0.8335）	目的网点_17	否（0.7018）

达的时长			
发车月份	10 月发车（0.6405）	车辆吨位	7 及以上（0.9344）
到达时刻	凌晨 0-5 点到达（0.7708）	目的网点_16	否（0.5403）
地区_6	从地区 6 出发（0.8503）	始发网点_18	否（0.7018）
卸货_小时	预计卸货 6 小时以上（0.7315）	车辆长度	9.6 及以上（0.9344）
线路编码	91,72,15,7,16,96 等（>=0.8）	地区_2	否（0.5656）
分拨_月份	9 月（0.6222）	地区_5	否（0.5403）
标的_月份	9 月（0.6223）	总里程	45-144（0.9344）
靠车_月份	10 月（0.6400）	目的地为河南	否（0.5403）
靠车时刻	凌晨 0-4 点停车（0.562）	始发网点_6	否（0.5474）

由表 4-1 可知，选取的十个重要变量与定价有一定的关联性，具体表现为：在对逻辑回归模型下，给定其他条件时，除靠车时刻在凌晨 0-4 点外，线路指导价较低的概率都大于 0.6，其中时间影响最为显著。XGBoost 模型中，只有目的网点不为 17、车辆吨位不小于 7、始发网点不为 18、车辆长度至少 9.6 以及总里程 4-144 的情况下，价格提高的概率大于 0.6。

基于以上的研究分析，本文可以得到会导致平台发布价格较低的四种情况：

(a) 预计卸货至少 6 小时以上、车的吨位和长度要求都表明当货物数量较多或者重量较大时；

(b) 发车到到达的市场和总里程数则表明当距离较远时；

(c) 91,72,15,7,16,96,8,2 等线路可能路况较差或者存在过路收费情况；

(d) 7 月及以后，尤其是 9、10 月份可能处于托运旺季，供给侧任务较多。

表 4-2 两个模型下的各个变量最大后验概率对应取值

定价较高（成交价/定价<0.95）			
逻辑回归		**XGBoost**	
变量	最大后验概率及其对应取值	变量	最大后验概率及其对应取值
从发车到到达的时长	不超过一个小时（0.376）	目的网点_17	是（0.6754）
发车月份	6 月发车（0.5646）	车辆吨位	3 及以下（0.6754）
到达时刻	下午一点至晚上 24（0.3548）	目的网点_16	否（0.3612）
地区_6	从地区 6 出发（0.2855）	始发网点_18	是（0.6754）
卸货_小时	预计卸货 6 小时以下（0.3427）	车辆长度	5.2 及以下（0.6754）
线路编码	85,86,89,88,83 等（>=0.6）	地区_2	否（0.3041）
分拨_月份	5 月（0.6172）	地区_5	否（0.3612）
标的_月份	5 月（0.6172）	总里程	小于 45（0.5913）
靠车_月份	6 月（0.5628）	目的地为河南	是（0.3612）
靠车时刻	下午 2 点以后（0.3446）	始发网点_6	是（0.5345）

同表 4-1 一样分析法一致，在表 4-2 中各变量与定价的关联性表现如下：逻辑回归模型中，线路编码为特定值、分拨和标的月份为 5 月时，价格调低的概率大于 0.6。在 XGBoost 模型中，当车辆较小和目的网点为 17 时，价格调低的概率大于 0.6.

基于以上的研究分析，本文可以得到会导致平台发布价格较高的四种情况：

(a) 85,86,89,88,83 等线路可能较为顺畅或者途径该线路的运输车辆较多；

(b) 5 月份发货较少或者运输车辆较多；

(c) 货物数量较少或者重量较小时；

(d) 从网点 18 出发和到达网点 17 的货运车可能较多。

4.4.2 基于 Kruskal-Wallis 检验的分析结果

在分析完选取的重要度前十的变量对线路指导价合适与否的影响后，本文还利用 Kruskal-WAllis 检验进行了统计分析分析，如果某个变量在不同组别下（C_1, C_2, C_3）的分布具有显著性差异（P<0.01），则认为它是造成第一次指导价没有达成交易的重要因素之一。Kruskal-Wallis 显著性分析结果如表 4-3。

表 4-3 基于 Kruskal-Wallis 分析各变量显著性

逻辑回归		**XGBoost**	
变量	P 值	变量	P 值
从发车到到达的时长	7.894×10^{-23}	目的网点_17	1.484×10^{-5}
发车月份	0.153	车辆吨位	6.870×10^{-10}
到达时刻	3.368×10^{-5}	目的网点_16	0.009
地区_6	0.309	始发网点_18	9.091×10^{-29}
卸货_小时	0.064	车辆长度	2.175×10^{-8}
线路编码	4.037×10^{-12}	地区_2	0.383
分拨_月份	1.575×10^{-6}	地区_5	0.026
标的_月份	5.710×10^{-14}	总里程	2.242×10^{-36}
靠车_月份	0.193	目的地为河南	0.738
靠车时刻	0.013	始发网点_6	0.456

从表 4-3 中我们可以看出，在逻辑回归筛选出的变量中，从发车到到达的时长、到达时刻、线路编码和分拨月份这 4 个变量在三种情况下分布是显著不同的，作出 4 个变量的频率分布直方图如图 4-3 所示。

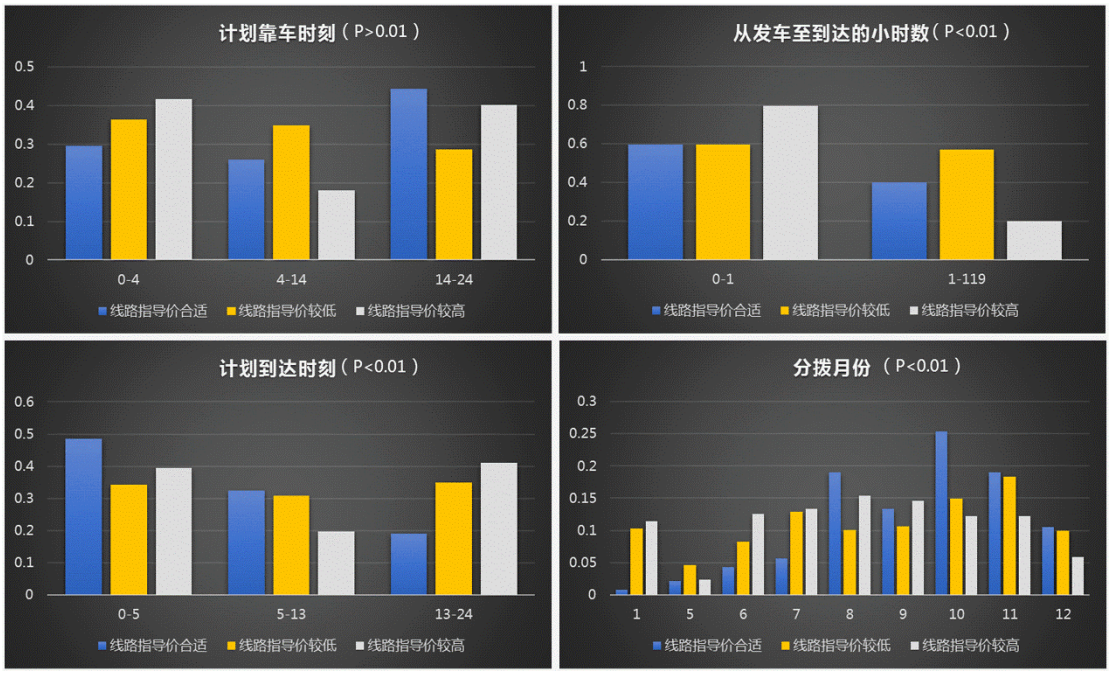

图 4-3 不同情况下具有显著性差异的变量分布（逻辑回归方法）

通过观察图 4-3 中 4 个变量的分布并结合统计结果我们得出以下结论：

（1）靠车时刻。停靠时间在晚上 20 点时线路指导价合适的数量是最多的；

（2）从发车到到达的时长。历时超过 1 小时更容易导致线路指导价较高；

（3）到达时刻。5-13 时比 0-5 时和 13-24 时定价较高的可能性大；

（4）标的月份与分拨月份。在 10 月份线路指导价合适的比例最高，11 月份次之；同时在 11 月份线路指导价较低的比例也是最高的，10 月份次之。导致这种现象原因可能是 10 月份、11 月份临近双 11，元旦等特殊时间，货主的需求较大且调价紧急程度高。

从表 4-3 中我们可以看出，XGBoost 筛选出的变量中，目的网点是否为 17、车辆吨位、目的网点是否为 16、始发网点是否为 18、车辆长度、总里程这 6 个变量在三种情况下分布显著不同，作出 6 个变量的频率分布直方图如图 4-4 所示。

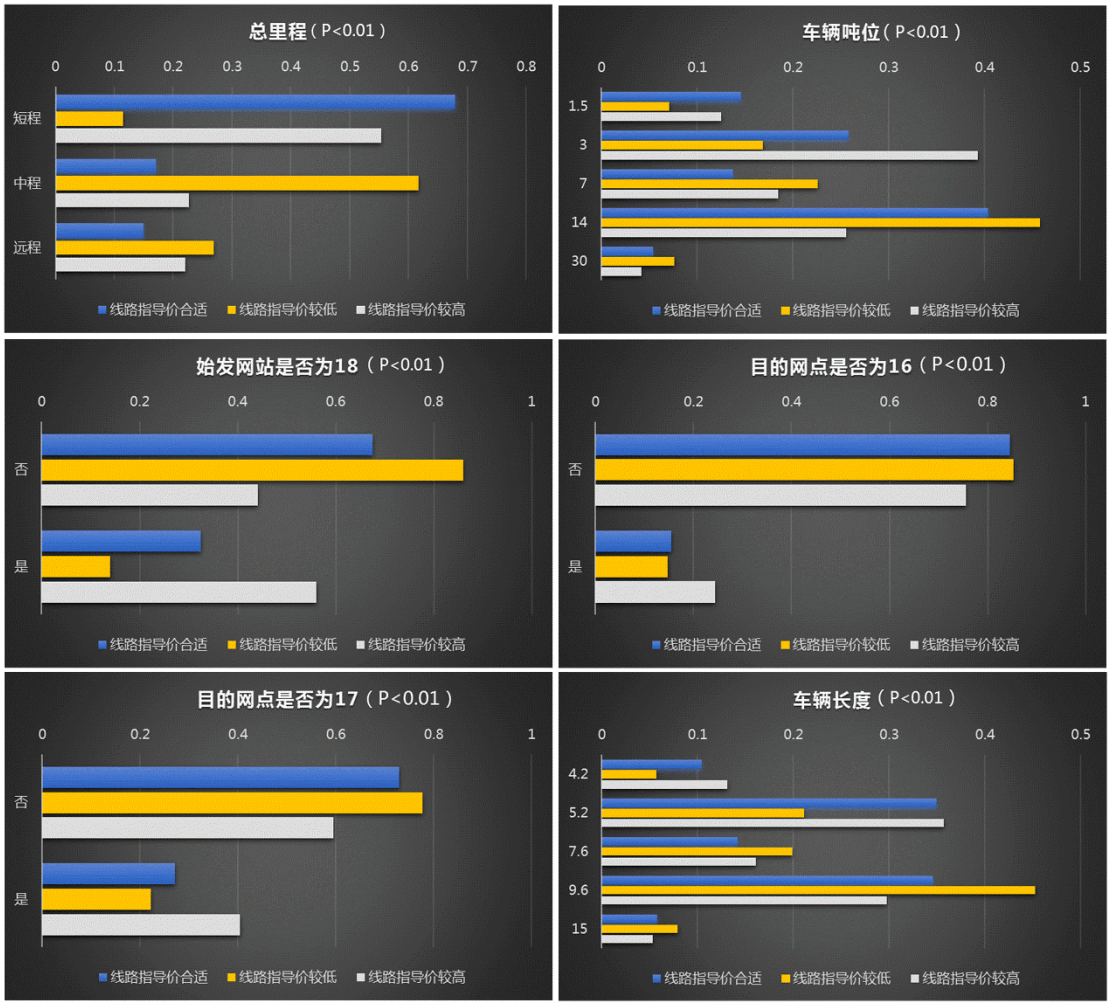

图 4-4 不同情况下具有显著性差异的变量分布（XGBoost 方法）

通过观察图 4-4 中 6 个变量的分布并结合统计结果我们得出以下结论：

（1）总里程。当总里程为中程和远程时，线路指导价较低的可能性大；然而在短程时，线路指导价合适或较高的可能性大。出现这种情况的原因可能是，中远程的定价本身就比短程的高，平台为例控制成本而给出相对较低的指导价，而承运端的司机不能接受；而短程较易成交，可能是因为愿意接受短程业务的司机较多，也就是说供是大于等于需的。

（2）车辆吨位。在三种类别下车辆的平均吨位为：9.239，10.887，7.497；众数分别为：14，14，3。鉴于此我们可以发现，当车辆的吨位比较低时，可能会导致线路指导价较高。

（3）始发网点是否为 18。始发网点在 18 的线路指导价较低的可能性小，而定价较高的可能性大。

（4）车辆长度。一车辆长度在三种类别下的平均长度分别为：7.529，8.391，7.289；众数分别为：5.2，9.6，5.2。当车辆长度为 9.6m 时，容易使得线路指导价定价较低。

5 问题三：模型建立与求解

5.1 问题分析与思路

根据题目要求可知，首次发布价格后最多还可刷新两次报价，并且平台以促进尽快成交和降低承运成本为主要目标。根据供求关系定价论可知运价取决于运输产品的供求关系，因此，在解决该问题之前，本文假设货源（货主）与车源（承运端司机）处于平衡状态。我们从以下两个角度对问题进行建模分析。首先，我们认为平台初次发布的价格，是综合考虑各项因素后能使平台获得期望利益或满足货主最高承运成本要求的期望报价，所以我们给出的第一次报价是以最低的承运成本为目标，而不关注成交率。其次，对第一次报价结束后尚未成交的任务，我们以提高成交率为目标，通过调价策略制定不同的定价，使得这些任务能快速成交。

因此，问题三的解答需要分为四步：

(1) 应用动态调价策略，基于弹性成交策略制定一个评价指标-成交率；

(2) 以附件 1 已成交数据的初次指导价为参考，延续平台的初次定价策略，制定第一次报价。

(3) 当第一次报价无法被承运端的司机认同时，则进入第二轮报价。基于附件 1 中的最终成交价建立回归模型，预测附件 2 中平台发布的初次指导价，作为第二次报价；

(4) 根据附件 1 可知，当成交价在平台报价的 95%和 105%之间时，则认为不需要调价。因此，对于附件 1 中的成交价，采用对抗训练思想的方法，建立回归模型预测第三轮报价，进一步增强模型泛化能力。

最后，对于总成本预测，采用同样建立有监督判别模型，基于附件 1 的总成本建立回归模型，预测附件二的总成本。

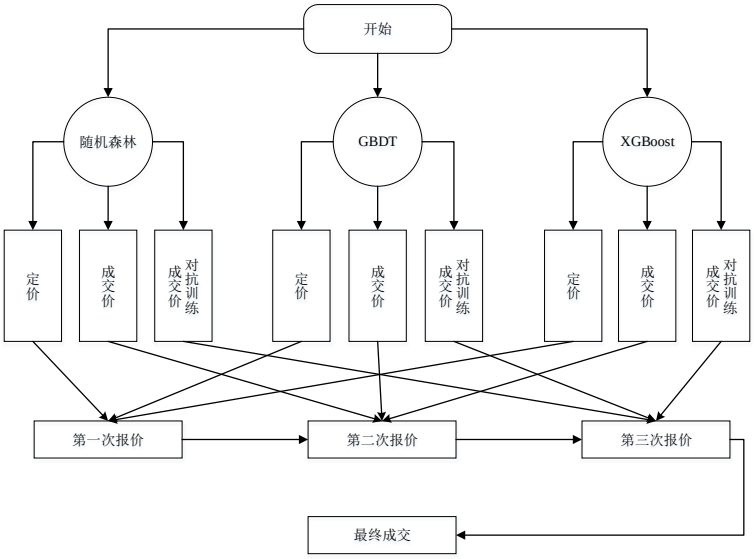

图 5-1 问题 3 求解思路

5.2 模型建立

5.2.1 定价策略评估指标

回归问题中，RMSE 只能评估预测值和真实值的偏差，而在本题中，RMSE 无法很好的衡量有多少任务在给出报价后是可以成交的。为了简单且合理的评估我们的定价策略，我们基于弹性成交策略制定了一个评价指标-成交率。对于某真实成交价 s，若我们的定价策略给出的报价 s_1 在真实成交价 s 的 95%到 105%之间，则认为该报价可以成交，并计算最终所有任务中成交的比例。成交与否计算公式如下：

$$deal = \begin{cases} 1, & s_1 \geq s \times 95\% \text{且} s_1 \leq s \times 105\% \\ 0, & \text{其他} \end{cases} \tag{5-1}$$

此处 deal 代表成交与否，deal 等于 1 时代表成交，deal 等于 0 时代表不成交。

5.2.2 基于集成学习的报价预测模型

为了同时保证较高的建模准确率和建模特征的合理性，我们摒弃了传统机器学习中复杂的特征工程，不使用特征交叉和特征组合等操作，以减少不可解释的建模特征。基于问题一所使用的建模特征之上，我们仅添加了对包括业务类型、线路编码和始发网点等在内的类别变量的频数统计作为建模特征，以进一步增强模型的准确性。

对每一次报价，都采用随机森林、GBDT 和 XGBoost 基于特定目标建立回归模型。对每个模型，综合其在 5 个不同随机数下的 5 折交叉验证结果，进一步提升模型的泛化能力，五折交叉验证过程如图 5-2 所示。得到每个模型的结果之后，采用 stacking 策略融合三个模型的结果。常见的模型融合策略包括 bagging，boosting，平均（加权）和 stacking。随机森林就是基于 bagging 策略建立的机器

学习模型，而 GBDT 和 XGBoost 都属于 boosting 家族经典的机器学习算法。由于平均法忽略了不同模型的准确性差异，加权法需要人为指定权重，因此我们最终采用 stacking 进行模型融合，以进一步提升模型的准确性和泛化能力。stacking 的第二层采用 LASSO 回归作为弱学习器，拟合目标标签和第一层模型书输出的特征，具体 stacking 策略如图 5-3 所示。

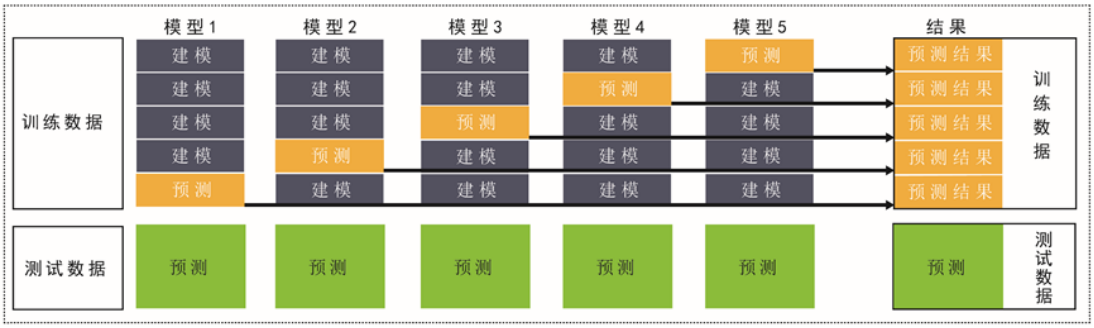

图 5-2 五折交叉验证过程

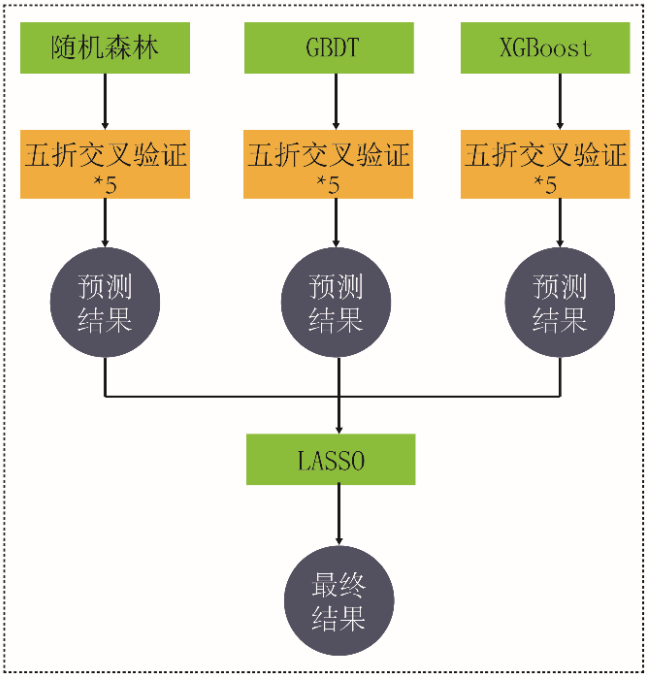

图 5-3 stacking 策略

5.3 模型求解

(1) 特征工程

在问题一所使用的特征基础上，加入了对各个类别变量包括业务类型、需求类型 1、需求类型 2、是否续签、标的展示策略、打包类型、运输等级、始发地省份名称、车辆类型分类、目的地省份名称、交易对象、地区类型、交易模式、需求紧急程度、地区、始发网点和目的网点的统计频率作为建模特征，在提升建

模精度的基础上保证了建模特征的合理性。

(2) 第一次报价

首先，以附件 1 中的线路指导价作为建模标签，以 5.3.1 中使用特征分别构建随机森林、GBDT 和 XGBoost 模型。对每个模型取其在 5 个不同随机数下（0,10,20,30,40,）5 折交叉验证结果的均值，得到每个模型对附件 1 中线路指导价的预测值和附件 2 中线路指导价的预测值。其次，基于 stacking 策略，用 LASSO 回归求解附件 2 最终线路指导价预测值。该最终指导价预测值即为第一次报价。

(3) 第二次报价

以附件 1 中的线路价格作为建模标签，以 5.3.1 中使用特征分别构建随机森林、GBDT 和 XGBoost 模型。对每个模型取其在 5 个不同随机数下（0,10,20,30,40,）5 折交叉验证结果的均值，得到每个模型对附件 1 中线路价格的预测值和附件 2 中线路价格的预测值。然后基于 stacking 策略，用 LASSO 回归求解附件 2 最终线路价格预测值。该最终线路价格预测值即为第二次报价。

(4) 第三次报价

早在 2014 年，Szegedy et al. [7]发现只要对深度学习模型的输入添加一些微小的扰动就能轻易改变模型的预测结果。后续的研究将该种扰动称之为对抗扰动，扰动后的输入称为对抗样本，将输入对抗样本误导模型的这一过程称为对抗攻击。深度学习模型遭遇对抗攻击时所表现出的脆弱性，给实际应用带来了极大的风险。自然语言处理的应用比如文本分类、情感分类、问答系统、推荐系统等也都受到了对抗攻击的威胁[8]

考虑到最终成交价并非固定不变，而是在不同的时空间环境下随着交易双方的博弈有着一定变化。因此对于求解第三次报价，我们采用了对抗训练的策略。为真实标签添加一定的扰动，以期望模仿真实交易中价格的轻微波动。虽然这会导致训练误差提高，但增强了模型的泛化能力和与原始预测模型之间的差异性。

以附件 1 中的线路价格作为建模标签，在其基础上随机添加一定扰动，使标签值在原始值的（95%到 105%之间波动），再采用类似于第二次报价中的方法求解扰动后的附件 1 线路价格预测值以及第三次报价。

(5) 总成本

采用上述同样方法求解附件 2 中的总成本预测值。

5.4 定价策略

我们的定价策略主要包括两部分，首先是为了尽可能的降低承运成本，满足供货商的需求。随后两次报价则是为了满足快速成交的要求，使得任务能以双方都认同的价格快速成交。

主要建模方法为机器学习回归预测模型和对抗训练，即基于经验数据挖掘出

各建模特征与线路指导价和线路成交价之间的数学映射，并基于该数学映射求解未知数据的线路指导价和线路成交价作为我们给出的报价。同时采用了 5 折交叉验证和 stacking 保证模型的准确性和泛化能力，采用对抗训练模拟价格的轻微波动以进一步提升模型的泛化能力。具体三次报价的调整策略如下：

（1）拟合历史数据线路指导价，求解新任务的线路指导价。若成交，怎能最大限度的满足供货商所期望的承运成本。

（2）拟合历史数据成交价，求解新任务的线路成交价，以保证第二次报价的准确性，促使快速成交。

（3）考虑价格的波动性。在对抗训练的基础上，拟合历史数据成交价，求解新任务的线路成交价，是对前两次报价的补充和完善，以期望在第三次报价时促进更多任务成交。

5.5 定价策略评估

考虑到 RMSE 无法很好的评价我们制定的报价策略带来的成交情况。因此我们采用成交率评估我们的报价策略。基于 5 折交叉验证，我们能在保证没有数据泄露的情况下，求得附件 1 的线路成交价预测值，对比该预测值与真实成交价，即可求得成交率，如表 5-1 所示。

表 5-1 报价成交率

模型	第一次报价成交率	第二次报价成交率	第三次报价成交率	综合第一二次报价成交率	综合三次报价成交率
随机森林	21.64%	95.68%	95.48%	96.13%	96.26%
GBDT	21.70%	95.39%	92.78%	95.90%	96.15%
XGBoost	21.69%	95.76%	95.64%	96.39%	96.53%

由表 5-1 可知，基于附件一平台报价预测所得的第一次报价的成交率只有 21%左右，这意味着绝大部分价格是不被司机认可的。而我们给出的第二次报价，可以让 95%以上的任务成交。综合对抗训练所得第三次报价，可让最终成交率达到 96%以上，即使在真是成交价的 99%到 101%区间之内，经过我们的模型给出的三次报价，也可达到至少 90%以上的成交率。

6 建议信

随着“互联网+”时代的到来，货运物流逐渐面向数字化和智能化方向发展。“无车承运人”模式通过资源共享，积极调动并促进车源与货源的信息匹配，可以进一步规范城市运输市场。通过本文的研究，在此向该平台就线路指导价问题作出如下建议：

第一，通过分析题目给定的附件 1 具体参数信息，构建了基于集成学习的思路的动态报价预测模型。对于附件 3 任务路线的 3 次定价参数，可以看出基于附件 1 给定的参数信息制定的总成本定价普遍会高于第一次报价，考虑到该平台处于试点阶段，第一次的报价可以考虑作为“赔本赚吆喝”的营销手段，可认为该本文的报价研究模型具有合理性。

第二，由于本文是基于供需平衡的角度建立的线路指导价模型，因此，为了能够保证货主和承运端司机数量处于在稳定平衡并持续增长状态，平台可以增加一个评价服务后台，对给予加入平台满 1 年以上并且评价结果好的货主一定比例的折扣和承运司机一定比例的提成；

第三，由于季节时间对线路的定价影响相对较大，平台可以考虑在运营淡季，利用优惠的价格，促进这季的运单成交率；对于运营高峰期，例如特殊时间段，货主的需求较大且调价紧急程度高，可以适当提高价格，赚取一定比例的利润保证平台的基本支出费用。

第四，平台高估了小批量货物的线路指导价，应当合理评估货物和车的匹配关系，对于较小货运车，可以给更低的报价；

第五，由于地区的差异化会导致交通运输量的不同，对始发地和目的地较偏远的地方因交通不便利，可以适当提高线路指导价；

第六，由于总里程和对线路的指导价也有很大的影响，因此，可以从总里程的角度出发，结合地区因素制定每公里具体的货运线路指导价；

第七，对部分线路编码为诸如 85,86,89,88,83 等的线路，平台可适当降低线路指导价，对 91,72,15,7,16,96,8,2 等线路，则应当提高线路指导价；

第八，对于从网点 18 出发，到达网点 17 的运输任务，平台应当降低线路指导价。

最后，本文通过动态调控价格的手段引导货主需求，与承运端的有限资源相匹配，为无车承运人平台的线路定价策略提供依据。

7 模型评价

7.1 模型优点

问题 1 中，基于多种机器学习中的算法思路建立预测线路指导价的回归模型，同时获得各个模型中变量重要度的和模型均方根误差。再基于均方根误差对各模型的变量重要度进行加权，得到最终的主要影响因素。该方法虽然步骤多但可以保证较高的精度。

问题 2 中，基于逻辑回归和 XGBoost 建立分类模型，筛选两个模型中重要度排在前十的变量。两个模型五折交叉验证后的 F1 值分别为 85%和 98%，这说明我们所建立的模型具有极大的可靠度。最后分析导致定价较低或者较高的原因。

问题 3 种，对于第三次报价采用对抗训练的思想，建立泛化能力更强的回归模型。基于我们的动态定价策略，第一次报价为了降低承运成本仅有 21%的成交率，第二次报价后成交率达到 95%，综合第三次对抗训练给出的报价能使三次报价后达到 96%以上的成交率，极大了提高了货运订单的成交率。

7.2 模型缺点

第一，特征工程比较简单，经过复杂的特征工程之后精度还可进一步提升。

第二，由于时间原因，没有使用多层感知机建模，加入非决策树类的异构模型，能使模型融合后的准确性大幅度提升。

第三，基于机器学习方法构建的模型大多属于黑箱机智，很难定量的给出定价的标准。

参考文献

[1]李刚,梁家卷,潘建新,彭小令,田国梁.多元统计分析及其应用[J].中国科学:数学,2020,50(05):571-584.

[2] Robert Tibshirani. Regression Shrinkage and Selection Via the LASSO[J]. Journal of the Royal Statistical Society: Series B (Methodological),1996,58(1).

[3]姚登举,杨静,詹晓娟.基于随机森林的特征选择算法[J].吉林大学学报(工学版),2014,44(01):137-141.

[4] Breiman L I , Friedman J H , Olshen R A , et al. Classification and Regression Trees (CART)[J]. Biometrics, 1984, 40(3):358.

[5] Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine[J]. Institute of Mathematical Statistics,2001,29(5).

[6] Chen T , Guestrin C . XGBoost: A Scalable Tree Boosting System[J]. 2016.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in Proceedings of the International Conference on Learning Representations, 2014.

[8] Wang, W., Wang, L., Tang, B., Wang, R., & Ye, A. (2019). A survey on Adversarial Attacks and Defenses in Text, 1–13. Retrieved from http://arxiv.org/abs/1902.07285

附 录（软件及代码）

A. 使用的软件

文本使用的软件有 Excel、SPSS、python

B. 编码方案

表 1 类别变量的编码方案

序号	因素	一般性描述	编码
1	业务类型	速运	1
		重货	2
2	需求类型 1	普通	1
		区域发运	2
		二程接驳	3
3	需求类型 2	计划	1
		临时	2
4	是否续签	非续签	1
		续签 ECP 审批驳回或撤销	2
		已分拨续签	3
5	标的展示策略	DIR	1
		BDC	2
6	打包类型	周期流向	1
		人工	2
		单边	3
		周期往返	4
		往返	5
7	异常状态	正常	1
8	运输等级	一级运输	1
		二级运输	2
		三级运输	3
9	始发地省份名称	广东省	1
		北京市	2
		河南省	3
		陕西省	4
10	车辆类型分类	厢式运输车	1
11	目的地省份名称	广东省	1
		北京市	2
		河南省	3
		新疆维吾尔自治区	4
12	交易对象	B	1
		BC	2
		B	3
13	地区类型	分拨区	1
		业务区	2
14	异常状态	正常	1

15	交易模式	抢单	1
16	需求紧急程度	常规订单	1
		特急订单	2
		紧急订单	3

C. 问题 1 四个模型的各类别变量的重要度

表 2 四个模型的各类别变量的重要度

LASSO(RMSE=391.0)		**随机森林**(RMSE=47.7)		**XGBoost**(RMSE=44.6)		**GBDT**(RMSE=42.3)	
因素	重要度	因素	重要度	因素	重要度	因素	重要度
总里程	0.9375	车辆吨位	0.1517	是否陕西出发	0.3906	运输等级是否为第一季	0.2496
始发网点是否为 12	0.037	车辆长度	0.1458	是否为一级运输	0.241	车辆长度	0.1898
车辆长度	0.0228	是否陕西省发车	0.1423	目的地是否为新疆	0.1526	目的地是否为新疆	0.1551
地区是否为 6	0.0164	是否为一级运输	0.1261	车辆长度	0.1108	总里程	0.1452
是否河南省出发	0.0162	地区是否为 4	0.1261	车辆吨位	0.0724	地区是否为 4	0.098
当天考车的时刻	0.0143	目的地是否为新疆	0.1035	总里程	0.0127	车辆吨位	0.0846
靠车到发车的时长	0.0143	总里程	0.0936	地区是否为 4	0.0113	是否陕西出发	0.0481
一个月中第多少天标的	0.014	发车到到达的时长	0.0884	发车到到达的时长	0.0039	发车到到达的时长	0.0104
分拨到靠车的时长	0.0126	目的网点是否为 20	0.0032	是否河南发出	0.001	第几个月分拨	0.0034
车辆吨位	0.01	第几月分拨	0.0028	地是否为 5	0.001	第几个月标的	0.0026
第几月分拨	0.0098	第几月标的	0.0027	目的地是否为河南	0.0004	一个月中第几天标的	0.0024
需求类型是否为有计划	0.0083	一个月中第几天标的	0.0015	目的网点是否为 10	0.0004	交易对象是否为 B	0.0018
是否为已分拨续签	0.0078	一个月中第几天分拨	0.0015	始发网点是否 19	0.0003	一个月中第几天分拨	0.0016
一个月中第多少天发车	0.0073	线路编码	0.0013	是否广东发出	0.0003	线路编码	0.0015
是否北京市	0.0064	交易对象是否为 B	0.001	目的地是否为广东	0.0002	到达时的时刻	0.0011
非续签	0.0055	第几个月发车	0.0008	是否北京发出	0.0002	第几个月发车	0.0007
续签 ECP	0.0055	计划卸货	0.0007	第几个月	0.0001	打包类型是	0.0006

审批驳回或撤销		等待时长		分拨		否为人工	
一天中什么时刻分拨	0.0052	第几个月靠车	0.0007	交易对象是否为 B	0.0001	业务类型是否为重货	0.0006
一个月中第多少天靠车	0.0051	是否为人工打包	0.0006	始发网点是否为 12	0.0001	靠车到发车的时长	0.0004
需求类型 1 是否为普通	0.0049	标的到分拨的时长	0.0004	始发网点是否为 11	0.0001	始发网点是否为 11	0.0003

D. 使用的代码

问题 1：特征工程

```
import pandas as pd
import numpy as np

def fillna(alld):
    for col in alld.columns:

        if col == '是否续签':
            alld[col] = alld[col].replace(['未知'],['N'])
        if 'N' in alld[col].value_counts().index:
            print(col)

            vc = alld[col].value_counts().sort_values(ascending=False)
            mode = vc.index[0] if vc.index[0]!='N' else vc.index[1]

            alld[col]    = alld[col].replace('N',mode)
    return alld

def obj2float(alld):

    alld['业务类型'] = alld['业务类型'].map({'速运':1,'重货':2})
    alld['需求类型 1'] = alld['需求类型 1'].map({'普通':1,'区域发运':2,'二程接驳':3})
    alld['需求类型 2'] = alld['需求类型 2'].map({'计划':1,'临时':2})
    alld['是否续签'] = alld['是否续签'].map({'非续签':1,'续签 ECP 审批驳回或撤销':2,
                                            '已分拨续签':3})
    alld['标的展示策略'] = alld['标的展示策略'].map({'DIR':1,'BDC':2})
    alld['打包类型'] = alld['打包类型'].map({'周期流向':1,'人工':2,
                                            '单边':3,'周期往返':4,'往返':5})
    alld['异常状态'] = alld['异常状态'].map({'正常':1})
    alld['运输等级'] = alld['运输等级'].map({'二级运输':1,'三级运输':2,'一级运输':3})

    alld['始发地省份名称'] = alld['始发地省份名称'].map({'广东省':1,'北京市':2,
```

```
                                                '河南省':3,'陕西省':4})
    alld['车辆类型分类'] = alld['车辆类型分类'].map({'厢式运输车':1,'北京市':2,
                                                '河南省':3,'陕西省':4})

    alld['目的地省份名称'] = alld['目的地省份名称'].map({'广东省':1,'北京市':2,
                                                '河南省':3,'新疆维吾尔自治区':4})

    alld['交易对象'] = alld['交易对象'].map({'B':1,'BC':2,'C':3})
    alld['地区类型'] = alld['地区类型'].map({'分拨区':1,'业务区':2})
#     alld['异常状态'] = alld['异常状态'].map({'分拨区':1,'业务区':2})
#     alld['异常状态'] = alld['异常状态'].map({'分拨区':1,'业务区':2})
    alld['交易模式'] = alld['交易模式'].replace(['抢单'],[1])
    alld['需求紧急程度'] = alld['需求紧急程度'].map({'常规订单':1,'特急订单':2,'紧急订单':3})

    return alld

def deal_time(alld):

    ### 众数填充缺失值后，计算相邻时间的差值，然后计算累计多少小时
    times = ['计划卸货完成时间','计划到达时间','计划发车时间','计划靠车时间',
             '分拨时间','标的_创建日期']
    time_diffs,hour_list,month_list,day_list = [],[],[],[]
    for i in range(len(times)-1):
        t1,t2 = pd.to_datetime(alld[times[i]]),pd.to_datetime(alld[times[i+1]])
        hour_list+=[t1.dt.hour.values.reshape(-1,1),t2.dt.hour.values.reshape(-1,1)]
        month_list+=[t1.dt.month.values.reshape(-1,1),t2.dt.month.values.reshape(-1,1)]
        day_list+=[t1.dt.day.values.reshape(-1,1),t2.dt.day.values.reshape(-1,1)]

        t_diff = t1-t2

        hours = t_diff.astype('timedelta64[D]').astype(int)*24+\
                        t_diff.astype('timedelta64[h]').astype(int)

        time_diffs.append(hours.values.reshape(-1,1))

    time_diffs = pd.DataFrame(np.hstack(time_diffs),
                              columns = ['卸货_到达_时间','到达_发车_时间',
                                         '发车_靠车_时间','靠车_分拨_时间',
                                         '分拨_标的_时间']

    ### 得到当天的小时（几点）
    hour_idx = [0,2,4,6,8,9]
    hour_list = np.hstack(hour_list)[:,hour_idx]
```

```
    hour_list = pd.DataFrame(hour_list,
                             columns = ['卸货_小时','到达_小时',
                                        '发车_小时','靠车_小时',
                                        '分拨_小时','标的_小时'])
        ### 得到几号
    day_idx = [0,2,4,6,8,9]
    day_list = np.hstack(day_list)[:,day_idx]

    day_list = pd.DataFrame(day_list,
                             columns = ['卸货_天','到达_天',
                                        '发车_天','靠车_天',
                                        '分拨_天','标的_天'])

        ### 得到月份
    month_idx = [0,2,4,6,8,9]
    month_list = np.hstack(month_list)[:,month_idx]

    month_list = pd.DataFrame(month_list,
                             columns = ['卸货_月份','到达_月份',
                                        '发车_月份','靠车_月份',
                                        '分拨_月份','标的_月份'])
    return time_diffs,hour_list,day_list,month_list

def drop_cols(alld):
    times = ['计划卸货完成时间','计划到达时间','计划发车时间','计划靠车时间',
             '分拨时间','标的_创建日期','标的_创建时间','计划发车日期']
    single_val = ['异常状态']
    alld.drop(times+single_val,axis=1,inplace=True)
    return alld

def get_alld():
    train = pd.read_excel('../数据/附件 1：货运线路历史交易数据.xlsx')
    test = pd.read_excel('../数据/附件 2：待定价的货运线路任务单.xlsx')
    test.drop(['编号'],axis=1,inplace=True)
    use_cols = test.columns

    ### 缺失值太多
    null_cols = ['子包号','装卸的次数','装的次数','卸的次数','是否需要装卸']
    use_cols = set(use_cols)-set(null_cols)
    target = ['线路总成本','线路价格（不含税）','线路指导价（不含税）']

    alld = pd.concat([train[list(use_cols)+target],test[use_cols]],axis=0).reset_index(drop=True)

    alld =  fillna(alld)
```

```
    float_alld = obj2float(alld)
    time_diffs,hour_list,day_list,month_list = deal_time(float_alld)

    float_alld = drop_cols(float_alld)

    all_data = pd.concat([float_alld,time_diffs,
                          hour_list,day_list,month_list],axis=1)

    all_data.to_csv('../数据/all_data.csv',index=0)
get_alld()
```

问题 1：多模型下的变量重要度

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor as rfg
from sklearn.linear_model import LASSO as LASSO
from sklearn.preprocessing import StandardScaler as ss
from XGBoost.sklearn import XGBRegressor as xgb
from sklearn.ensemble import GradientBoostingRegressor as gbr

def onehot(data):
    obj_cols = ['业务类型','需求类型 1','需求类型 2','是否续签','标的展示策略',
                '打包类型','运输等级','始发地省份名称','车辆类型分类',
                '目的地省份名称','交易对象','地区类型',
                '交易模式','需求紧急程度','地区','始发网点','目的网点']

    onehot_list = []
    for col in obj_cols:
        onehot_list.append(pd.get_dummies(data[col],prefix=col))

    oh_list = pd.concat(onehot_list,axis=1)
    data.drop(obj_cols,axis=1,inplace=True)
    data = pd.concat([data,oh_list],axis=1)
    return data

def load_data():
    data = pd.read_csv('../数据/all_data.csv')
    data.drop(['任务 id'],axis=1,inplace=True)

    data = onehot(data)

    label_cols = ['线路价格（不含税）', '线路总成本',
                  '线路指导价（不含税）']

    labels = data[label_cols]
    tr_size = labels.notnull().sum()[0]
```

```
    tr_y = labels['线路指导价（不含税）'][:tr_size]

    tr_x = data.drop(label_cols,axis=1).iloc[:tr_size,:]
    cols = tr_x.columns

    tr_x = pd.DataFrame(ss().fit_transform(tr_x))
    return tr_x,tr_y,cols

def get_importance(tr_x,tr_y,cols):

    reg_rf = rfg(n_estimators=300,max_depth = 15,
                                 max_features=0.6,random_state = 2020)
    reg_LASSO = LASSO(alpha=2)

    reg_gbdt = gbr(
                                 learning_rate = 0.1,
                                 n_estimators = 100,
                                 max_depth = 12,
                                 max_leaf_nodes = 90,
                                 random_state = 2020
                                 )
            reg_xgb = xgb(learning_rate =0.01,
                                  n_estimators=1200,
                                  max_depth=12,
                                  gamma=0.4,
                                  colsample_bytree= 0.5,
                                  seed = 2020)

    model_list = [reg_rf,reg_LASSO,reg_gbdt,reg_xgb]
    model_names = ['rf','LASSO','gbdt','xgb']

    for model,name in zip(model_list,model_names):
        clf = model
        clf.fit(tr_x,tr_y.values)

        if name == 'LASSO':
            feat_imp = pd.Series(clf.coef_,index=cols)
        else:
            feat_imp = pd.Series(clf.feature_importances_,index=cols)

        feat_imp = feat_imp.map(lambda x: round(abs(x)/sum(feat_imp.values),4))
        feat_imp = feat_imp.sort_values(ascending=False)
        feat_imp.to_csv('../结果文件/问题 1_%s_imp.csv'%(name))
```

```
tr_x,tr_y,cols = load_data()
get_importance(tr_x,tr_y,cols)
```

问题 2：三分类问题建模，获取变量重要度

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler as ss
from XGBoost.sklearn import XGBClassifier as xgb
from sklearn.linear_model import LogisticRegression as lr

def onehot(data):
    obj_cols = ['业务类型','需求类型 1','需求类型 2','是否续签','标的展示策略',
                '打包类型','运输等级','始发地省份名称','车辆类型分类',
                '目的地省份名称','交易对象','地区类型',
                '交易模式','需求紧急程度','地区','始发网点','目的网点']

    onehot_list = []
    for col in obj_cols:
        onehot_list.append(pd.get_dummies(data[col],prefix=col))

    oh_list = pd.concat(onehot_list,axis=1)
    data.drop(obj_cols,axis=1,inplace=True)
    data = pd.concat([data,oh_list],axis=1)
    return data

data = pd.read_csv('../数据/all_data.csv')
data.drop(['任务 id'],axis=1,inplace=True)
data = onehot(data)
label_cols = ['线路价格（不含税）', '线路总成本',
        '线路指导价（不含税）']
labels = data[label_cols]
tr_size = labels.notnull().sum()[0]
price_ratio = labels['线路价格（不含税）'][:tr_size]/labels['线路指导价（不含税）'][:tr_size]
idx0 = price_ratio[(price_ratio<=1.05) & (price_ratio>=0.995)].index
idx1 = price_ratio[(price_ratio>1.05)].index
idx2 = price_ratio[(price_ratio<0.995)].index
price_ratio[idx0] = 0
price_ratio[idx1] = 1
price_ratio[idx2] = 2
tr_y = price_ratio
tr_x = data.drop(label_cols,axis=1).iloc[:tr_size,:]
cols = tr_x.columns
tr_x = pd.DataFrame(ss().fit_transform(tr_x))
res_prob = np.squeeze(np.zeros((tr_x.shape[0],1)))
```

```
reg_xgb = xgb(
            objective = 'multiclass',
            learning_rate =0.1,
            n_estimators=100,
            max_depth=12,
            gpu_id = 0,
            colsample_bytree= 0.5,
            seed = 2020)

reg_lr = lr()

#model = [reg_lr,'lr']
model    = [reg_xgb,'xgb']

clf = model[0]
if model[1]=='xgb':
      feat_imp = pd.Series(clf.feature_importances_,index=cols)
else:
      feat_imp = pd.Series(clf.coef_[0],index=cols)
feat_imp = feat_imp.map(lambda x: round(abs(x)/sum(feat_imp.values),4))
feat_imp = feat_imp.sort_values(ascending=False)
print(feat_imp)
feat_imp.to_csv('../结果文件/第二问%s_imp.csv'%(model[1]))
#第二问，基于上一步的重要度生成新的数据
import pandas as pd
from XGBoost.sklearn import XGBClassifier as xgb
import random
from sklearn.linear_model import LogisticRegression as lr
import numpy as np

def onehot(data):
      obj_cols = ['业务类型','需求类型 1','需求类型 2','是否续签','标的展示策略',
                  '打包类型','运输等级','始发地省份名称','车辆类型分类',
                  '目的地省份名称','交易对象','地区类型',
                  '交易模式','需求紧急程度','地区','始发网点','目的网点']

      onehot_list = []
      for col in obj_cols:
            onehot_list.append(pd.get_dummies(data[col],prefix=col))

      oh_list = pd.concat(onehot_list,axis=1)
      data.drop(obj_cols,axis=1,inplace=True)
      data = pd.concat([data,oh_list],axis=1)
```

```
    return data

def random_sample(data,use_cols,n):
    samples = []
    for i in range(n):
        sample = []
        for col in use_cols:
            vc = data[col]
            idx = random.sample(list(range(len(vc))),1)[0]

            sample.append(vc[idx])

        samples.append(sample)

    return pd.DataFrame(samples,columns = use_cols)

def lr_xgb_random_sample():
    random.seed(2020)
    data = pd.read_csv('../数据/all_data.csv')
    data = onehot(data)
    lr_imp = pd.read_csv('../结果文件/第二问_lr_imp.csv',header=None,index_col=0)
    lr_use_cols = list(set(lr_imp.index[:10]))
    rand_sam_lr = random_sample(data,lr_use_cols,1000)


    label_cols = ['线路价格（不含税）', '线路总成本',
            '线路指导价（不含税）']
    labels = data[label_cols]
    tr_size = labels.notnull().sum()[0]
    price_ratio = labels['线路价格（不含税）'][:tr_size]/labels['线路指导价（不含税）'][:tr_size]
    idx0 = price_ratio[(price_ratio<=1.05) & (price_ratio>=0.995)].index
    idx1 = price_ratio[(price_ratio>1.05)].index
    idx2 = price_ratio[(price_ratio<0.995)].index
    price_ratio[idx0] = 0
    price_ratio[idx1] = 1
    price_ratio[idx2] = 2
    tr_y = price_ratio

    tr_x = data[lr_use_cols].iloc[:tr_size,:]
    clf = lr()
    clf.fit(tr_x,tr_y)
    pred = clf.predict_proba(rand_sam_lr)
    pred = np.argmax(pred,axis=1)
    rand_sam_lr['label'] = pred
```

```
    rand_sam_lr['label'].replace([0,1,2],['一次成交','价格被调高','价格被调低'],inplace=True)
    rand_sam_lr.to_csv('../数据/问题二待分析数据_lr.csv',index=0)

    xgb_imp = pd.read_csv('../结果文件/第二问_xgb_imp.csv',header=None,index_col=0)
    xgb_use_cols = list(set(xgb_imp.index[:10]))
    rand_sam_xgb = random_sample(data,xgb_use_cols,1000)
    tr_x = data[xgb_use_cols].iloc[:tr_size,:]
    clf = xgb(
                            objective = 'multiclass',
                            learning_rate =0.1,
                             n_estimators=100,
                             max_depth=12,
                             gpu_id = 0,
                             colsample_bytree= 0.5,
                             seed = 2020)

    clf.fit(tr_x,tr_y)
    pred = clf.predict_proba(rand_sam_xgb)
    pred = np.argmax(pred,axis=1)
    rand_sam_xgb['label'] = pred
    rand_sam_xgb['label'].replace([0,1,2],['一次成交','价格被调高','价格被调低'],inplace=True)
    rand_sam_xgb.to_csv('../数据/问题二待分析数据_xgb.csv',index=0)

    clf.fit(tr_x,tr_y)
    pred = clf.predict_proba(rand_sam_xgb)
    pred = np.argmax(pred,axis=1)
    rand_sam_xgb['label'] = pred
    rand_sam_xgb['label'].replace([0,1,2],['一次成交','价格被调高','价格被调低'],inplace=True)
    rand_sam_xgb.to_csv('../数据/问题二待分析数据_xgb.csv',index=0)

lr_xgb_random_sample()

#第二问，计算后验概率
from collections import Counter as counter
import numpy as np
import pandas as pd

class Mean_encoder():
    def __init__(self):
        pass

        def sort_dict_value(self,dic):
        ### 对标签按其数量进行降序排列，计算后延概率时，选取样本数量较多的 N-1 的进行计算
        return dict(sorted(dic.items(), key=lambda x:x[1], reverse=True))
```

```
    def valcount2prob(self,dic):
        prob = dict()
        for key,value in dic.items():
            prob[key] = value/sum(dic.values())
        return prob

    def fit(self,x,y):
        self.x, self.y = x,y
        self.postprob_records = []
        x_train, y_train = self.x, self.y
        y_train = np.squeeze(y_train)
        assert ( (y_train.size == y_train.shape[0]) and (len(y_train.shape)==1) )
        y_valcount = self.sort_dict_value(counter(y_train))
        self.prior_prob = self.valcount2prob(y_valcount)
        post_prob = {}
        for y_value in list(self.prior_prob.keys()):
            print(y_value)
                ##对特征进行遍历
            col_dict = {}
            for col in range(x_train.shape[1]):
                feat = np.squeeze(x_train[:,col])
                assert len(feat.shape)==1
                feat_valcount = self.sort_dict_value(counter(feat))
                    ##对特征值进行遍历，计算后延概率
                single_feat_dict = {}
                for featval,valcount in feat_valcount.items():
                    y_idx = np.where(y_train==y_value)
                    featval_yval_count = len(np.where(x_train[y_idx,col]==featval)[0])
                    ##  记录单个特征各个取值的后验概率和 shrinkage parameter
                    single_feat_dict[featval] = featval_yval_count/valcount
                    ##  记录各列特征的后验概率和 shrinkage parameter
                    col_idx_name = 'col'+str(col)
                    col_dict[col_idx_name] = single_feat_dict
                ##记录不同 y 取值下的后验概率和 shrinkage parameter
                post_prob[y_value] = col_dict
            ##记录不同 cv 下的后验概率和 shrinkage parameter
        return post_prob

def onehot(data):
    obj_cols = ['业务类型','需求类型 1','需求类型 2','是否续签','标的展示策略',
                '打包类型','运输等级','始发地省份名称','车辆类型分类',
                '目的地省份名称','交易对象','地区类型',
                '交易模式','需求紧急程度','地区','始发网点','目的网点']
```

```
    onehot_list = []
    for col in obj_cols:
        onehot_list.append(pd.get_dummies(data[col],prefix=col))
    oh_list = pd.concat(onehot_list,axis=1)
    data.drop(obj_cols,axis=1,inplace=True)
    data = pd.concat([data,oh_list],axis=1)
    return data
lr_imp = pd.read_csv('../结果文件/第二问_lr_imp.csv',header=None,index_col=0)
use_cols = list(set(lr_imp.index[:10]))
data = pd.read_csv('../数据/all_data.csv')
data = onehot(data)
label_cols = ['线路价格（不含税）', '线路总成本',
        '线路指导价（不含税）']
labels = data[label_cols]
tr_size = labels.notnull().sum()[0]
price_ratio = labels['线路价格（不含税）'][:tr_size]/labels['线路指导价（不含税）'][:tr_size]
idx0 = price_ratio[(price_ratio<=1.05) & (price_ratio>=0.995)].index
idx1 = price_ratio[(price_ratio>1.05)].index
idx2 = price_ratio[(price_ratio<0.995)].index
price_ratio[idx0] = 0
price_ratio[idx1] = 1
price_ratio[idx2] = 2
tr_y = price_ratio
tr_x = data[use_cols].iloc[:tr_size,:]

xgb 变量后验概率
'''
#code = pd.qcut(tr_x['总里程'].values,4)
#tr_x['总里程'] = pd.qcut(tr_x['总里程'].values,4).codes

逻辑回归变量后验概率
'''
tr_x['到达_发车_时间'] = pd.qcut(tr_x['到达_发车_时间'].values,3,duplicates='drop').codes

tr_x['卸货_小时'] = pd.qcut(tr_x['卸货_小时'].values,3,duplicates='drop').codes
tr_x['到达_小时'] = pd.qcut(tr_x['到达_小时'].values,3,duplicates='drop').codes
tr_x['靠车_小时'] = pd.qcut(tr_x['靠车_小时'].values,3,duplicates='drop').codes

me = Mean_encoder()
post_prob = me.fit(tr_x.values,tr_y.values)
print(post_prob)
```

问题 3：获取预测值，三个标签（对抗训练）

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor as rfg
```

```
from sklearn.metrics import mean_squared_error as mse
import numpy as np
from XGBoost.sklearn import XGBRegressor as xgb
from sklearn.ensemble import GradientBoostingRegressor as gbr
from sklearn.model_selection import KFold as SKFold
import random
from collections import Counter

def onehot(data):
    obj_cols = ['业务类型','需求类型 1','需求类型 2','是否续签','标的展示策略',
                '打包类型','运输等级','始发地省份名称','车辆类型分类',
                '目的地省份名称','交易对象','地区类型',
                '交易模式','需求紧急程度','地区','始发网点','目的网点']

    onehot_list = []
    for col in obj_cols:
        onehot_list.append(pd.get_dummies(data[col],prefix=col))

    oh_list = pd.concat(onehot_list,axis=1)
    data.drop(obj_cols,axis=1,inplace=True)
    data = pd.concat([data,oh_list],axis=1)
    return data

def count_feat(data):
    count_feats = []
#     month_cols = [col for col in data.columns if '月份' in col]

    obj_cols = ['业务类型','需求类型 1','需求类型 2','是否续签','标的展示策略',
                '打包类型','运输等级','始发地省份名称','车辆类型分类',
                '目的地省份名称','交易对象','地区类型',
                '交易模式','需求紧急程度','地区','始发网点','目的网点','线路编码']
    for col in obj_cols:
        count_feats.append(data[col].map(dict(Counter(data[col]))))

    count_feats = pd.concat(count_feats,axis=1)
    count_feats.rename(columns = dict(zip(count_feats.columns,
                                          [col+'_count' for col in obj_cols])),inplace=True)
    return count_feats

def N_cv_get_submission(model,train_x,train_y,test_x,seed=20,cv=10):
    kfold = SKFold(cv,random_state = seed, shuffle=True)

    cv_score = []
    train_record = np.zeros((train_x.shape[0],1))
```

```
    test_record = np.zeros((test_x.shape[0],cv))

    for i,(train_idx,test_idx) in enumerate(kfold.split(train_x,train_y)):
        cv_train_x, cv_train_y = train_x.iloc[train_idx,:], np.squeeze(train_y.values)[train_idx]
        cv_test_x, cv_test_y = train_x.iloc[test_idx,:], np.squeeze(train_y.values)[test_idx]
        clf = model[0]
        clf_name = model[1]

        if clf_name == 'xgb':
            clf.fit(cv_train_x,cv_train_y,
                    eval_set=[(cv_train_x, cv_train_y), (cv_test_x, cv_test_y)],
                    eval_metric='rmse',
                    verbose=500,
                    early_stopping_rounds=50)
            cv_train_res = clf.predict(cv_test_x)
            test_res = clf.predict(test_x)

        else:
            clf.fit(cv_train_x,cv_train_y)
            cv_train_res = clf.predict(cv_test_x)
            test_res = clf.predict(test_x)
        train_record[test_idx,:] = cv_train_res.reshape(-1,1)
        print('cv%s'%(i),np.sqrt(mse(cv_test_y,cv_train_res)))
        cv_score.append(np.sqrt(mse(cv_test_y,cv_train_res)))
        test_record[:,i] = test_res

    return train_record,test_record,cv_score

random.seed(2020)
data = pd.read_csv('../数据/all_data.csv')
data.drop(['任务 id'],axis=1,inplace=True)
count_feats = count_feat(data)
data = onehot(data)
label_cols = ['线路价格（不含税）', '线路总成本',
        '线路指导价（不含税）']
dic = dict(zip(label_cols+['noise'],['第二次报价','总成本','第一次报价','第三次报价']))
data = pd.concat([data,count_feats],axis=1)

labels = data[label_cols]
tr_size = labels.notnull().sum()[0]
'''
target = '线路价格（不含税）' or '线路总成本' or '线路指导价（不含税）'
'''
target = '线路价格（不含税）'
```

```
tr_y = labels[target][:tr_size]
'''
是否添加扰动，对抗训练
'''
noise = False
if noise:
    tr_y = pd.Series([round(random.uniform(x*0.95,x*1.05),0) for x in tr_y.values])
tr_x = data.drop(label_cols,axis=1).iloc[:tr_size,:]
cols = tr_x.columns
te_x = data.drop(label_cols,axis=1).iloc[tr_size:,:]
tr_seeds = np.zeros((tr_x.shape[0],1))
te_seeds = np.zeros((te_x.shape[0],1))

for seed in [0,10,20,30,40]:
    a = [12]
    b = [1]
    for depth in a:
        for num in b:
            reg_xgb = xgb(learning_rate =0.03,
                          n_estimators=1000,
                          max_depth=12,
                          gamma=0.4,
                          colsample_bytree= 0.4,
                          seed = 2020)

            reg_rf = rfg(n_estimators=300,max_depth = 18,
                          max_features=0.6,random_state = 2020)

            reg_gbdt = gbr(
                          learning_rate = 0.1,
                          n_estimators = 100,
                          max_depth = 12,
                          max_leaf_nodes = 90,
                          max_features= 42,
                          random_state = 2020
                          )

            #model    = [reg_rf,'rf']
            #model    = [reg_gbdt,'gbdt']
            model    = [reg_xgb,'xgb']
            train_res,test_res,cv_score= N_cv_get_submission(model,tr_x,tr_y,te_x,seed = seed, cv=5)
            te_seeds += np.mean(test_res,axis=1).reshape(-1,1)
            tr_seeds += train_res.reshape(-1,1)
            f1 = np.sqrt(mse(tr_y,train_res))
```

```
                print(depth,num,f1,seed,'\n')
clf = model[0]
feat_imp = pd.Series(clf.feature_importances_,index=cols)
feat_imp = feat_imp.map(lambda x: round(abs(x)/sum(feat_imp.values),4))
feat_imp = feat_imp.sort_values(ascending=False)
test_res = pd.DataFrame(te_seeds/5,columns=['pred'])
train_res = pd.DataFrame(tr_seeds/5,columns=['pred'])
if noise:
    train_res.to_csv('../数据/%s_train_%s_5%.csv'%(dic['noise'],model[1]),index=0)
    test_res.to_csv('../数据/%s_test_%s_5%.csv'%(dic['noise'],model[1]),index=0)
else:
    train_res.to_csv('../数据/%s_train_%s_5%.csv'%(dic[target],model[1]),index=0)
    test_res.to_csv('../数据/%s_test_%s_5%.csv'%(dic[target],model[1]),index=0)

#第三问，在上一步输出的基础上，做 stacking
import pandas as pd
from sklearn.linear_model import LASSO
import os

data = pd.read_csv('../数据/all_data.csv')
label_cols = ['线路价格（不含税）', '线路总成本',
        '线路指导价（不含税）']
labels = data[label_cols]
tr_size = labels.notnull().sum()[0]
tr_y = labels['线路价格（不含税）'][:tr_size]
file_path = '../数据/'
files = os.listdir(file_path)
'''
target = '总成本' or '第一次报价' or '第二次报价' or '第三次报价'
'''
target = '总成本'
files = [x for x in files if target in files]
tr_files = sorted([ i for i in files if 'train' in i])
te_files = sorted([ i for i in files if 'test' in i])

trs = []
for file in tr_files:
    trs.append(pd.read_csv(file_path+file))
trs = pd.concat(trs,axis=1)

tes = []
for file in te_files:
    tes.append(pd.read_csv(file_path+file))
tes = pd.concat(tes,axis=1)
```

```
reg = LASSO().fit(trs,tr_y)
pred = reg.predict(tes)

pred = pd.DataFrame(pred,columns=['pred'])
pred.to_csv('../结果文件/%s.csv'%(target),index=0)
```