队伍编号	203717
题号	A

基于回归模型和随机森林的无车承运平台动态定价研究

摘 要

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。本文针对无车承运平台的定价与后续调价的确定问题，基于相关性分析、回归分析和随机森林算法的特征重要性筛选出相互不相关的对于定价影响大的变量。接着，我们通过溢价率和自我构建的综合评定指标进行了传统的定价模型进行评价。最后，我们通过随机森林模型对于指导价和线路总成本进行预测，并通过非线性回归进行了动态调价。

对于无车承运平台定价的主要因素分析的问题，由于平台的承运订单信息数据存在维度高、数据类型多、变量间相关性高的特点，因此，在进行数据进行数值化处理和筛选以后，我们首先通过相关性分析的方式，剔除了相关性较高的变量。接着，我们通过线性回归的方式，计算了各个相互独立变量的对于定价的系数和 p 值。同时，我们还通过随机森林计算了各个变量的特征重要性(feature importance)。综合两个模型，我们发现总里程、车辆长度、运输等级、业务类型对于整体定价的影响度很高。同时我们发现根据总里程很明显得分成两堆：小于500公里的短距离订单和大于2000公里得长距离订单。同样运用这两个模型，我们发现对于小于 500 公里的短距离运输，车辆长度对于货运指导价也具有一定的影响；而对于大于 2000 公里的长距离运输，交易对象、是否续签、需求紧急程度以及打包类型对于定价具有一定的影响。

对于无车承运平台的定价评价模型的问题，我们首先引入金融中的溢价率的计算评定方法发现这个模型效果不佳同时溢价率高的数据是因为业务类型的原因而导致的溢价率的高。然而，因为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，溢价率只能代表较低的承运成本，所以我们构建了自己的成本时效综合指标：通过最终定价、指导价、交易成功时长和需求紧急程度来进行模型评定。

对于无车承运平台的定价策略模型，我们首先通过随机森林算法通过总里程，车辆长度，业务类型，运输等级，需求紧急程度、打包类型这六个影响度最重要的变量对于线路总成本和指导价进行预测，并通过五折交叉验证的方法进行判断准确性的验证：线路总成本模型准确率是 99.26%;指导价的准确率是 98.36%。最后，我们还用溢价率模型和成本时效综合指标进一步验证，效果比传统定价模型的结果好。对于无车承运平台的调价策略模型，我们通过交易对象的反馈次数与调价比例的分布规则，构建了非线性回归模型。结果表明，对于承运商来说，第一次到第三次的调价比例分别为 92%,129%和 166%，对于个体司机来说，调价比例分别为 89%,114%,140%。

根据上述结论，我们给无车承运平台写了建议信，并对于该研究进行了优缺点的分析，提出了相应的提高方向。

关键词： 回归模型 溢价率 随机森林算法

Research on Dynamic Pricing of Car-Free Transportation Platform Based on Regression Model and Random Forest

Abstract

With the gradual rise of the carless carrier industry in China, the scientific pricing of the carrier line is an urgent problem to be solved by many carless carrier platforms. In this paper, aiming at the determination of pricing and subsequent pricing adjustment of car-free transport platform, unrelated variables with great impact on pricing are screened out based on correlation analysis, regression analysis and random forest algorithm. Then, the traditional pricing model through premium rate and self-constructed comprehensive evaluation index are evaluated. Finally, the guide price and total line cost are predicted by random forest model, and the price is adjusted dynamically by nonlinear regression.

For the main factor analysis of the pricing of the car free carrier platform: because the carrier order information data of the platform has the characteristics of high dimension, multiple data types and high correlation between variables. Therefore, after the numerical processing and screening of the data, we first eliminate the variables with high correlation by means of correlation analysis. Then, we calculate the pricing coefficient and P value of each independent variable by linear regression. At the same time, we also calculate the feature importance of each variable through random forest. Combining the two models, we find that the total mileage, vehicle length, transportation level and business type have a high impact on the overall pricing. At the same time, we found that according to the total mileage, it is obviously divided into two piles: short-distance orders less than 500 kilometers and long-distance orders greater than 2000 kilometers. Similarly, using these two models, we find that for short-distance transportation less than 500 km, the vehicle length also has a certain impact on the freight guide price; For long-distance transportation greater than 2000 kilometers, the trading partner, renewal or not, demand urgency and packaging type have a certain impact on the pricing.

For the pricing evaluation model of car free carrier platform: firstly, we introduce the calculation and evaluation method of premium rate in finance. It is found that the effect of this model is poor, and the data with high premium rate is the high premium rate due to the original business type. However, at the current stage, the focus of the platform is to quickly promote the transaction and lower the carrier cost, and the premium rate can only represent the lower carrier cost. Therefore, we have constructed our own comprehensive cost effectiveness index: model evaluation through final pricing, guidance price, transaction success duration and demand urgency.

For the pricing strategy model of the car free carrier platform: firstly, we use the random forest algorithm to predict the total line cost and guide price through the six most important variables: total mileage, vehicle length, business type, transportation level, demand urgency and packaging type, The accuracy of the total line cost model is 99.26%; The accuracy of the guidance price is 98.36%. Finally, we further verify it with premium rate model and cost effectiveness comprehensive index, and the effect is better than that of traditional pricing model. For the price adjustment strategy model of car

free carrier platform, we construct a nonlinear regression model through the distribution rules of feedback times and price adjustment proportion of trading partners. The results show that for carriers, the price adjustment rates from the first to the third are 92%, 129% and 166% respectively, and for individual drivers, the price adjustment rates are 89%, 114% and 140% respectively.

According to the above conclusions, we wrote a recommendation letter to the vehicle free carrier platform, analyzed the advantages and disadvantages of this research, and put forward the corresponding improvement direction.

Key words: regression model, premium rate, random forest algorithm

目录

一、问题重述

1.1 问题背景

自国内公路运输市场对个人开放以来，市场逐渐出现了“小，散，乱”的特点。为规范运输市场，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。高效的交易模式和低廉的承运成本是无车承运人试点平台持续健康发展的关键。因此，如何科学合理地制定交易指导价和调价策略，是目前众多无车承运人平台提升核心竞争力所亟待解决的问题。

1.2 问题重述

在本课题中，我们的目标是根据历史订单信息，以无车承运人的视角，以快速促进成交和较低的承运成本为目标，研究面向广大拥有运力资源（货车）的承运端司机的平台动态定价机制（包括初始定价和后期调价策略）。具体需要解决以下三个问题：

1. **数据清理和关键变量选取：**从订单信息中，我们会发现，货运的定价订单中存在很多变量，呈现高维特征，而且数据也存在大量缺失的情况。因此，我们需要对于数据进行清洗和处理，并将它转换成适合进行数据分析和挖掘的形式。同时，我们需要从中筛选出影响无车承运人平台进行货运的线路定价的主要因素，为后期定价体系建立奠定基础。
2. **定价评价指标构建：**基于筛选后的变量，我们要设计评价的目标函数并建立数学模型，对历史定价体系的效果进行评价，衡量历史价格策略的合适程度。
3. **线路定价和调价模型建立：**根据历史订单模型的数据，建立交易价格初始定价的预测模型和后期动态调价策略的预测模型，通过数据训练和交叉验证，尽可能提升预测的准确率。

二、问题分析

2.1 变量的清理和选取

根据给出的历史订单数据，我们发现数据存在维度高、表达形式多样、缺失值多和变量间相关性高的特点。因次，我们拟首先对数据集中的变量类型进行判断，按文字型变量和数字型变量两大类进行分析和标准化处理。然后，对于处理后的数据进行清洗，去除无效化数据列、异常数据行和缺失过多的数据列。

在完成数据清理之后，再利用相关性分析、线性分析、特征提取算法寻找主要影响因素（变量），通过特性重要度指标评估各个特征在影响无车承运人平台进行货运线路定价问题上的比重，并利用变量信息熵增益进行验证。

2.2 定价评价指标构建

第二问是根据成交货运线路历史报价和最后交易价格数据，对现有的定价模式进行评价。根据无车承运人试点平台“高效的交易模式和低廉的承运成本”的建设目标，价格评价应该从交易时长和价格吻合度这两个方面进行设计。由于提供的历史交易数据中缺乏没有成交的交易数据（提供的都是成交记录）且并没有调价次数的记录，因此无法将未成交情况作为评价的内容，对不成交情况进行惩罚。

从成本视角考量，我们拟借鉴金融交易的定价评定方法，采用了溢价率指标进行分

析成交价和最初定价的差异。从效率视角的考量，我们将综合考虑实际成交时长、货物运输的紧急程度的指标，分析交易效率和时间有效性。因此，我们需要建立了一个成本时效综合指标用来评判这个订单价格合理性。

2.3 线路定价预测模型建立

根据第三问的要求，我们需要建立关于线路定价预测的数学模型，根据目前数据的特点，根据问题一筛选的主要影响因素，我们预选了多个经典的预测类算法（BP 神经网络、支持向量机、随机森林），进行模型初步构建和效果检验，发现随机森林在预测准确度上具有显著优势。因此，以此算法为基础，进行模型的细化训练和分析。

2.4 调价预测模型建立

除了需要进行线路的初始定价研究，还需要对后期调价策略进行研究。由于目前的数据中缺乏同一任务若干次报价的历史数据，因此无法建立有效的时序数据进行调价预测，只能从与调价比例有关的变量上进行分析。通过关联分析，目前记录表内的变量实际上与调价都呈弱相关或不相关，仅仅与反馈次数有关，且数据呈现明显的散布特征。因此，我们将从调价比例和反馈次数之间寻找其内涵的统计规律，并由此建立调价预测模型。

三、基本假设

1. 定价的时候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）的承运端司机。
2. 无车承运人以一定价格提前发布到网络平台上供承运端司机浏览并决定是否承运该运输任务。
3. 平台的定价机制应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。
4. 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。

四、符号说明

符号	符号意义
$H(X)$	信息熵
$p_i(i=1,2,\cdots,n)$	集合中的变量 $X=\{x_1,x_2,\cdots,x_n\}$ 对应在集合的概率
$H(Y\mid X)$	信息的条件概率
$p_{最初}$	第一次的报价
$p_{实际}$	最终的成交价
$p_{需求}$	该订单的需求紧急程度，其中从正常到非常紧急分别是[0.25, 0.5, 0.75]
$t_{完成}$	最终的交易时长

五、问题一模型的建立与求解

5.1 数据预处理与筛选

5.1.1 数据清理

在订单的数据表中，存在数字型变量和文字型变量两种。在数字型变量中，时间变量在这个问题中相关性比较小，所以不进行纳入；为了方便分析，文字型变量通过不同的方法进行标准化转化，将这些变量变为数字型的分类变量，具体的处理方法如图 1 所示：

图 1 订单信息数据变量分类及处理方法

其中，对于多分类文字型变量采用是 one-hot 编码 （“独热编码”）进行处理。即：对于 N 个变量值用 N 位状态寄存器编码 N 个状态，每个状态都有独立的寄存器位，且这些寄存器位中只有一位有效，即只能有一个状态。例如：在交易对象中有个体司机、承运户以及两者皆有这三种情况，则其特征向量对应有[(0,1),(1,0),(1,1)]三种情况。

5.1.2 数据筛选与相关性分析

变量被统一为数字型变量后，进行初步数据清理并进行相关性分析。

首先，基于对于数字型变量数据集，首先需要清除具有高度耦合性的变量。例如：车辆长度与车辆吨位成强相关性如图 2：

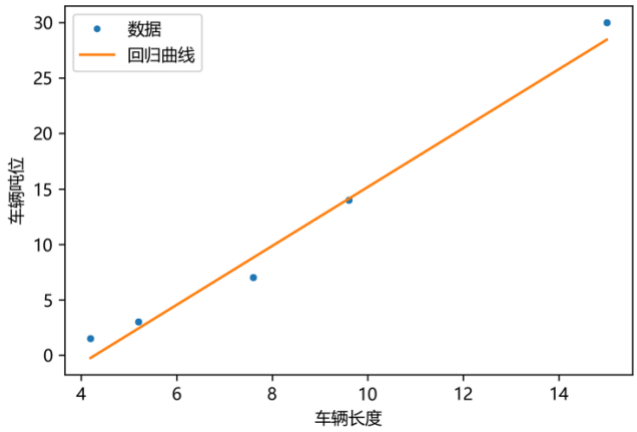

图 2 车辆长度与车辆吨位的关系

为了避免类似车辆长度与车辆吨位这类呈现较强的线性正相关（strong linear positive correlation）的影响分析效果，避免出现模型的共线性等问题，我们需要对变量进行解耦处理。

挑选车辆长度进行保留变量之后，我们建立相关性矩阵对所有变量的相关性系数进行计算，观察是否还存在类似车辆长度与车辆吨位的相关性高的变量，避免后期预测模型中共线性问题的产生（如图 3）。

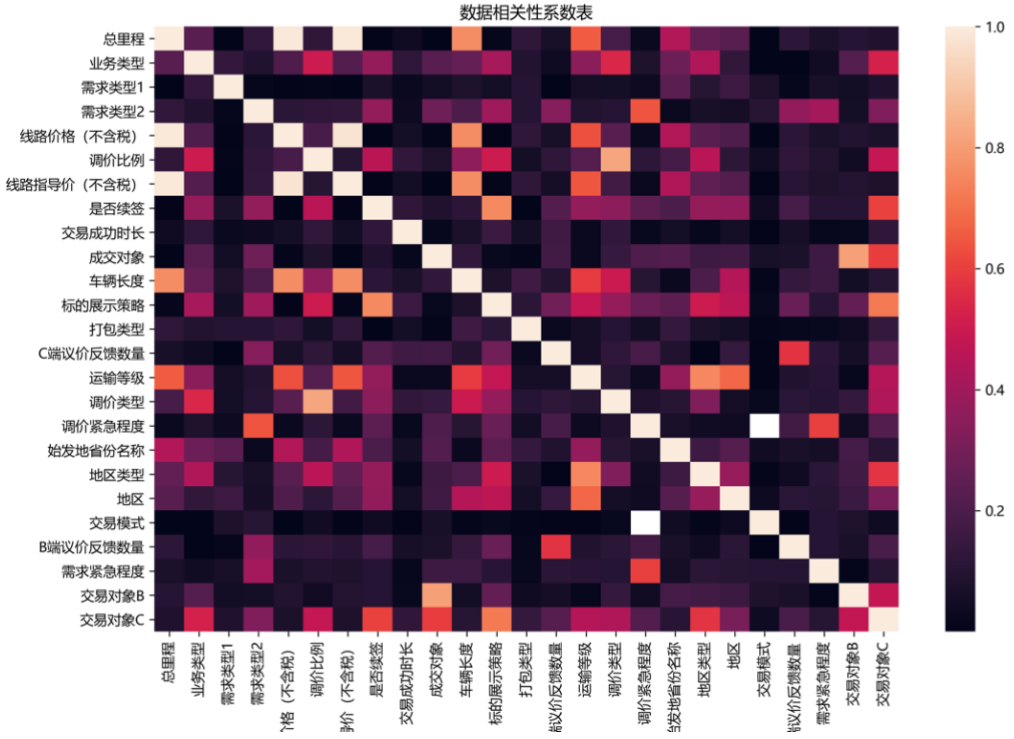

图 3　数据相关性分析图

因此，通过建立相关性矩阵，并结合相关性系数表，在筛选获得 14 个变量（见图 4 的纵轴变量列表），以避免因共线性而导致模型无法进行拟合或者拟合结果较差的情况出现。同时，我们发现：（1）总里程、车辆长度、成交对象以及运输等级，与平台给出的线路指导价格呈现较强的相关性；（2）交易模型、标的展示策略以及交易模式，与调价的紧急程度呈现较强的相关性。此两条发现，亦可作为参考结论与后文由模型得出的结论进行有效性验证对比，具有较强的参考价值。

5.2 线性回归模型分析

为了进一步判断影响无车承运人平台进行货运线路定价的主要因素、为后续预测模型提供更准确的变量选择方向，将相关性分析后筛选后的 14 个数字型变量全部输入多元线性回归模型之中作为自变量，而将货运线路指导定价作为多元线性回归模型中的因变量。

5.2.1 线性回归模型建立

在此问题中，因变量数据经过数据筛选与清理，由相关性分析所给出的变量之间的相关程度对于各个变量都是独立的，即可认为随机干扰项期望为零，故可采用多元线性回归模型。模型自变量包括：业务类型、成交对象、需求类型 1、需求类型 2、紧急程度、地区类型、交易对象 B、交易对象 C、车辆长度、总里程、标的展示策略、是否续签、打包类型以及运输等级，货运线路指导定价为因变量，建立多元线性回归模型。

5.2.2 线性回归模型结果分析

首先，由多元线性回归模型得到单个变量系数如图 4 所示：

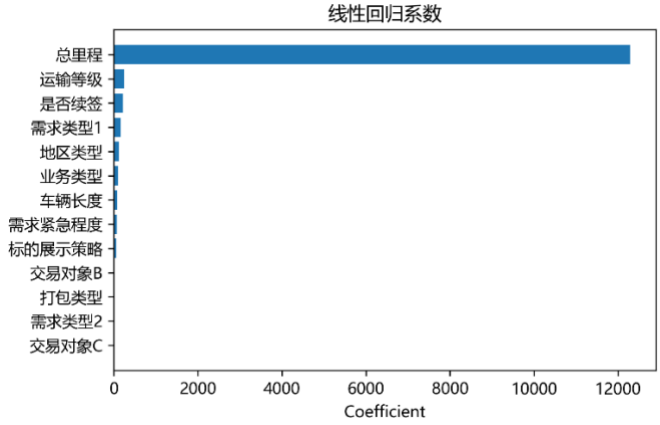

图 4 多元线性回归系数图

从图中清晰地得出，总里程对于自变量，即货运指导价的影响远超于其他因变量。为了更全面地研究各个变量，探究其他变量对于自变量的影响，暂时将总里程剔除出多元线性回归模型，得出如下单个变量系数图（剔除总里程）如图 5：

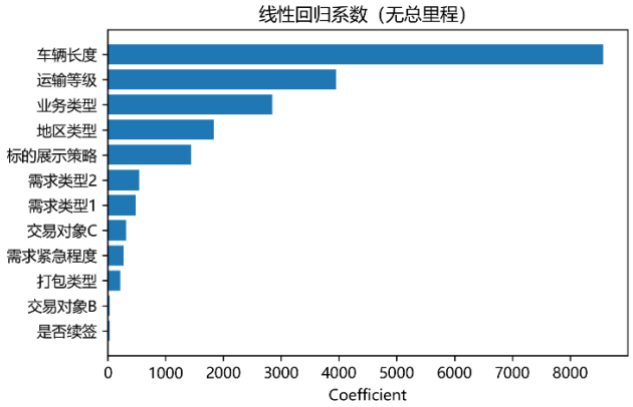

图 5 线性回归系数（无总里程）

我们发现车辆长度与运输等级相比于其他因变量对于自变量也具有较为显著的影响。此外，为了更好地消除总里程的压制性影响效果，我们绘制了总里程与定价的散点图：

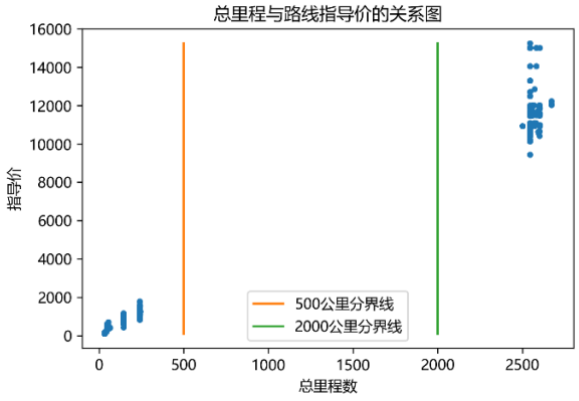

图 6 总里程与定价的关系

我们清晰地了解到数据集主要集中在总公里数小于 500 公里与大于 2000 公里的短途与长途运输区间内，且由散点图可清晰辨识出小于 500 公里的数据呈现出里程数与成交货运线路指导定价具有极为明显的线形关系；但是大于 2000 公里的数据呈现出长条形的数据云，并不能很好得印证里程数对于成交货运线路指导定价具有最强的线性相关性这一结论。

因此，我们将数据集按照总里程数分为小于 500 公里与大于 2000 公里的短途与长途运输区两个数据子集，并分别再次使用多元线性回归模型，得出图 7 和图 8 两张分别针对小于 500 公里的短途运输与大于 2000 公里的长途运输的单个变量系数图：

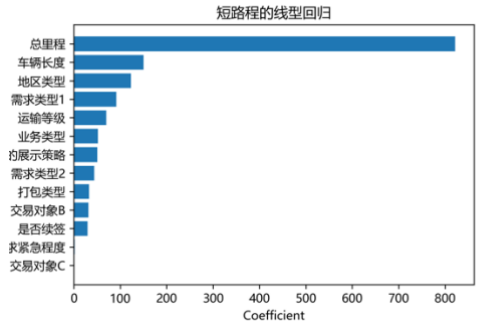

图 7 线性回归系数（短路程）

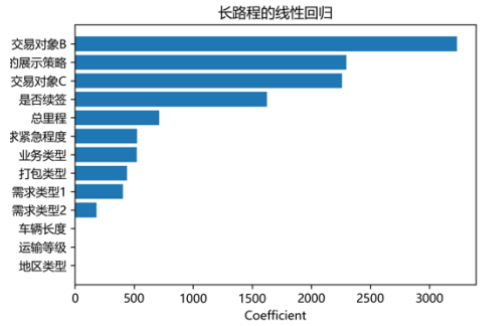

图 8 线性回归系数（长路程）

从图 7 和图 8，我们发现对于小于 500 公里的短途运输，相对于其他变量，总里程对于货运指导价具有压倒性的影响；而对于大于 2000 公里的长途运输，交易对象（B、C）以及标的展示策略这几个变量取代了总里程，展现出对于货运指导价的主要影响。

通过多元线性回归模型分别得到各个因变量在模型中调整之后对于自变量的 P 值见图 9。可见，需求程度 2 与打包类型的 P 值都已经高于 0.20，故在模型中自变量，即货运指导价格对于这两个因变量的解释性不强。

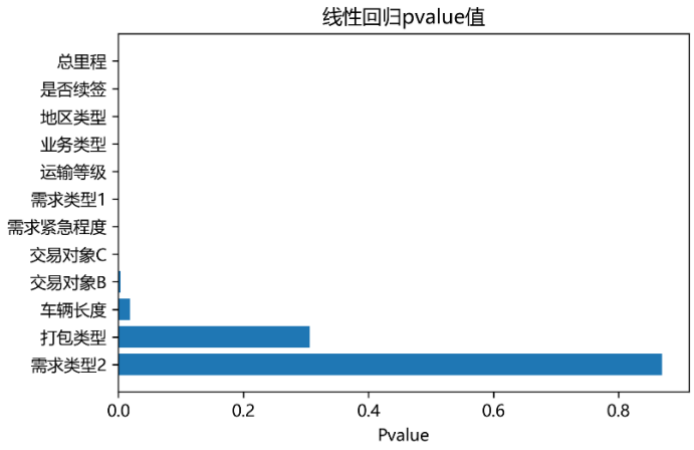

图 9 线性回归 p-value 值

总体看，总里程、车辆长度以及运输等级这三个变量对货运指导价的影响较为显著。在小于 500 公里的短途运输与大于 2000 公里的长途运输中，情况出现不同。对于小于 500 公里的短途运输，相对于其他变量，总里程对于货运指导价具有压倒性的影响；而对于大于 2000 公里的长途运输，交易对象以及标的展示策略取代了总里程，展现出对于货运指导价的主要影响。

5.3 基于随机森林特征的重要性分析

随机森林是一种机器学习模型（非线性基于树的模型）集成学习方法。随机森林算法具有在运算量没有显著提高前提下，预测精度高的优势。同时，随机森林对多元共线性不敏感，结果对缺失数据和非平衡数据比较稳健，可以很好地预测多达几千个解释变量的作用。

通过与神经网络、支持向量机算法的初算对比，结合本次数据的特点，为了能（1）有效地大数据集上运行来评估各个特征在分类问题上的重要性；（2）具有较高的准确率；（3）对于缺省值问题获得良好的结果；（4）避免内部生成误差，我们决定采用随机森林（Random Forest）算法来评估各个特征在影响无车承运人平台进行货运线路定价问题上的比重。利用模型获得各个变量的重要性，并借此评估各个变量的影响权重。

5.3.1 随机森林模型的建立

随机森林（图 10）通过自助法(Bootstrap)重采样技术，从原始训练样本集 N 中有放回地重复随机抽取 k 个样本(k 一般和 N 相同)生成新的训练样本集，然后根据自助样本集生成 n 个分类树组成随机森林。因此，我们可以通过随机地对袋外数据所有样本的特征 X 加入噪声干扰，来分析该特征的重要性。我们将上述 14 个变量作为特征，建立随机森林模型，分析每个特征对最后价格影响的重要度。[1]

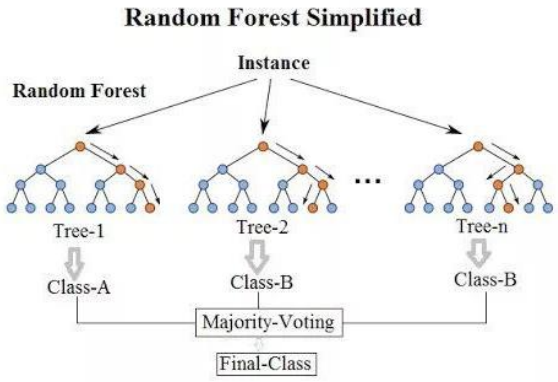

图 10 随机森林原理图

5.3.2 随机森林模型的结果分析

由随机森林模型得到单个变量的特征重要度（Feature importance）图如下：

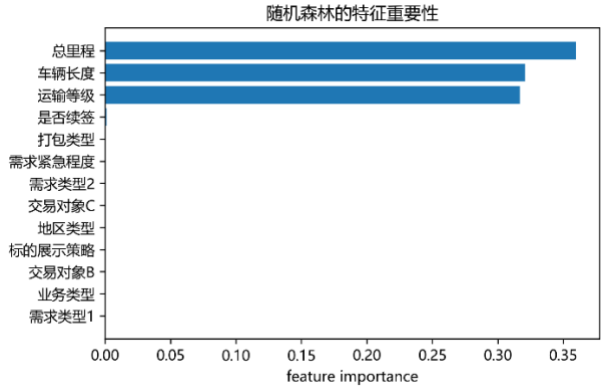

图 11 随机森林的特征重要性

其中，总里程的重要程度居于首位，而运输等级与车辆长度对于货运指导价也具有远超于其他变量的重要程度。

通过上文多元线性回归模型发现，在小于 500 公里的短途运输与大于 2000 公里的长途运输的两种情况中，各个变量对于货运指导价的影响作用存在不同的情况。因此，使用随机森林模型分别对于小于 500 公里的短途运输与大于 2000 公里的长途运输这两种情况进行进一步的分析评估，分别得出如下两张单个变量的特征重要度图：

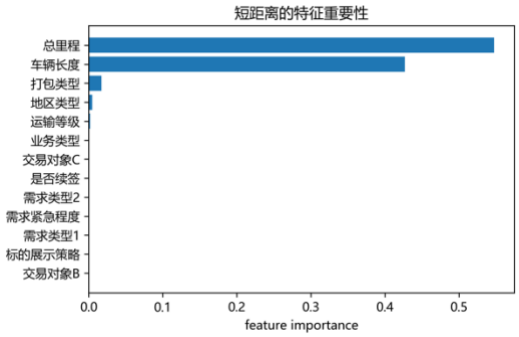

图 12 随机森林的特征重要性（短里程）

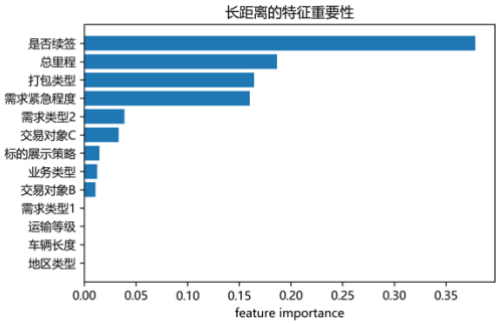

图 13 随机森林的特征重要性（长里程）

从上面的图中，我们发现对于小于 500 公里的短途运输，总里程与车辆长度对于货运指导价具有较强的影响；而对于大于 2000 公里的长途运输，相比于其他变量，是否续签取代了总里程，展现出对于货运指导价较强的影响。是否续签这一因素，是线性回归模型未发现的因素。

5.4 重要变量的验证

5.4.1 信息增益模型

为了进一步对重要变量进行验证，我们使用决策树算法中常用的信息增益方法来分析与验证上文所发现的相关性较高的变量是否对于定价具有较大的影响。对于待划分的数据集 D，信息增益是其划分前的熵（前）与基于某一特征变量划分之后的熵（后）差值。因此信息增益差异越大，说明如果使用当前特征划分数据集 D，则说明此属性被应用于划分数据集 D 效果较好，否则则认为该属性不适合用于划分数据集 D，该特征更有利于决策。这里信息增益 Ent_{div}，通过公式（1）进行计算：

$$H(x)=-\sum_{i=1}^{n} p_i \log_2(p_i)$$

$$H(Y\mid X)=\sum_{i=1}^{n} p_i(x)H(Y\mid X=x_i)$$

$$Ent_{div}=H(x)-H(Y\mid X)$$

其中，$H(X)$ 表示信息熵，$p_i(i=1,2,\cdots,n)$ 表示集合中的变量 $X=\{x_1,x_2,\cdots,x_n\}$ 对应在集合的概率，$H(Y\mid X)$ 表示信息的条件概率。[2]

单个变量的信息熵如下：

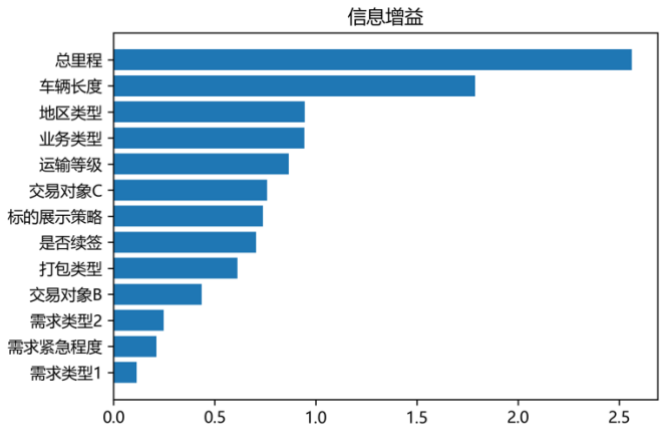

图 14 各变量的信息增益

我们可以明显看出，总里程与车辆长度这两个变量对于模型划分极为显著；而地区类型、业务类型以及运输等级这一类变量对于模型划分相对显著。结合上文的多元线性回归与随机森林模型，能够更加有助于进行决策。

5.4.2 结果与分析

结合上文的多元线性回归模型、随机森林模型特征选择并通过信息增益模型的验证，可以得到如下结论：

1）总里程、车辆长度、运输等级、业务类型对于定价的影响度是影响定价最为重要的主要因素。

2）小于 500 公里的短距离运输，需求类型以及标的表示策略两种变量对于货运指导价也具有一定的影响。

3）而对于大于 2000 公里的长距离运输，交易对象、是否续签、需求紧急程度以及打包类型对于货运指导价具有一定的影响。

综上，上述变量将用于后文提出的预测模型建立提供依据。

六、问题二模型的建立与求解

6.1 溢价率指标

6.1.1 指标设计

为了对于已经成交货运线路历史交易数据中的定价进行评价，我们首先引入了金融中的溢价率的概念。溢价率（公式 2）指的是在有限的剩余期限内，投资者要达到保本的目的，表的资产价格至少要想投资者有利方向变动的幅度。溢价率可以视为权衡权证风险高低的一个指标，溢价率越高，说明该定价模型对于无车承运人的风险越大，相反，溢价率的越低，说明风险越小，效果也越好。

$$F=\frac{(P_{成交}-P_{成本})}{P_{成本}}\times 100\% \qquad (2)$$

其中，F 表示溢价率，$P_{成交}$表示最终成交价，$P_{成本}$ 表示线路总成本。[3]

6.1.2 定价效果评价

根据公式（2），我们可以得到每一个订单的溢价率的数值，同时，我们可以从图 16

可以看出，溢价率分布有两个峰值，一种的是小于 0.3 的溢价率，这样的溢价率代表的是定价的效果相对来好，而另外一种是大于 0.3 的溢价率，这样的溢价率代表的是及定价的效果相对较差，总体来说，从这个结果，我们可以发现，原有的定价模型的定价效果一般。

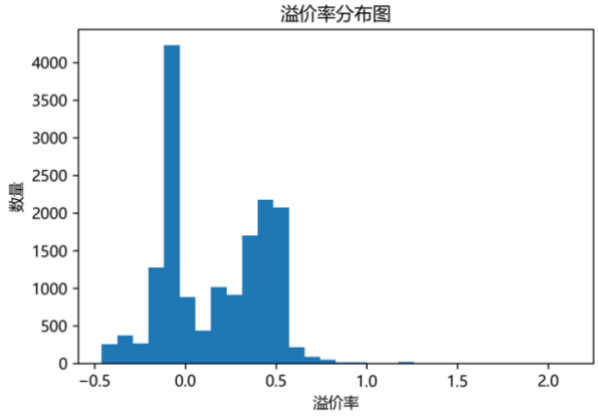

图 15 溢价率分布频数分布图

我们对影响溢价率的因素进行分析。通过对于各因素分类前与分类后的对比图（16-21），我们会发现业务类型对于定价的影响非常大，同时，表的展示策略和是否续签是影响定价效果的主要因素，在指导价中应该尽可能的对这些量进行分析。

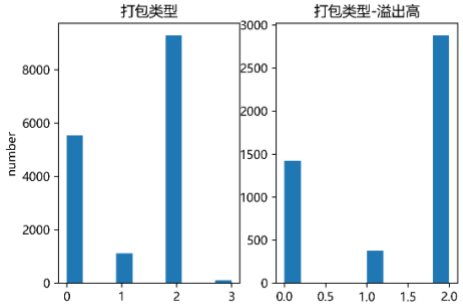

图 16 打包类型在溢价率高低中的对比分布图

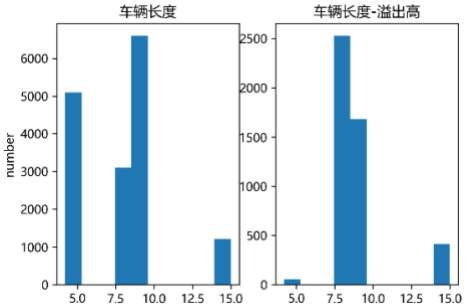

图 17 车辆长度在溢价率高低中的对比分布图

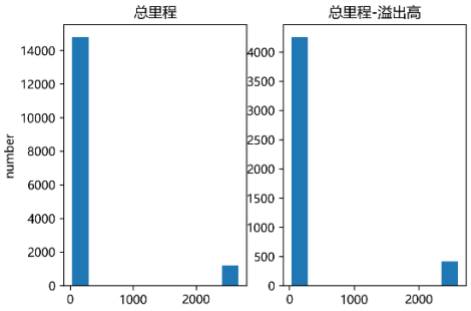

图 18 总里程在溢价率高低中的对比分布图

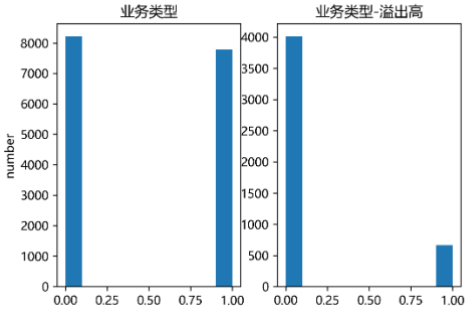

图 19 业务类型在溢价率高低中的对比分布图

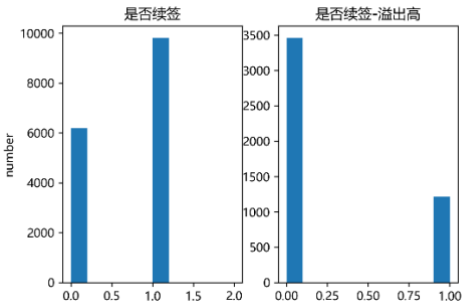

图 20 续签在溢价率高低中的对比分布图

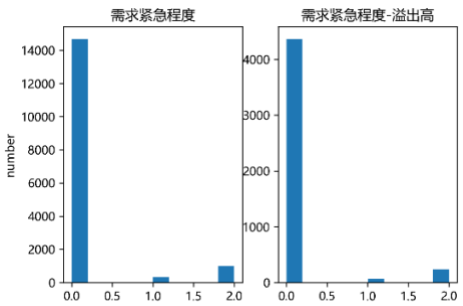

图 21 需求紧急程度在溢价率高低中的对比分布图

6.2 成本时效综合指标

6.2.1 指标设计

无车承运人平台建设目标是：较低承运成本的运输方案，还需要考虑快速促成承运订单达成。因此，我们需要自己设计要给综合评价指标，不仅能够评价他的价格变动，同时也可以评价他的快速性，最后，我们建立了如下的得分模型，见公式（3）。

$$E=(p_{最初}-p_{实际})\times(1-p_{需求})-t_{完成}\times p_{需求} \qquad (3)$$

其中，E 表示成本时效综合指标

$p_{最初}$ 表示归一化后的第一次的报价，

$p_{实际}$ 表示归一化后最终成交价，

$t_{完成}$ 表示的是最终的交易时长

$p_{需求}$ 表示的是该订单的需求紧急程度从正常到非常紧急分别是 $[0.25, 0.5, 0.75]$。

对变量进行归一化的目的是为了保证这个模型的量纲相同。对于成本，引入了溢价率中用最终定价与成本求差值的概念，并结合需求的紧急程度，可以反映寻找需求紧急程度和成本之间互相牵制，达到一个最优的平衡。指标值的计算范围在$[-1, 1]$，指标值越小，说明该订单定价的效果差，而若这个指标值越大，说明该订单的定价效果好。指标值为0，说明定价基本合理。

6.2.2 定价效果评价

从图 22 计算结果我们可以看出，原有的定价指导价策略还是相对来说，其时效性表现比较稳定的，总体平均值略大于 0，没有特别多指标偏小的值，但表现不够优异。

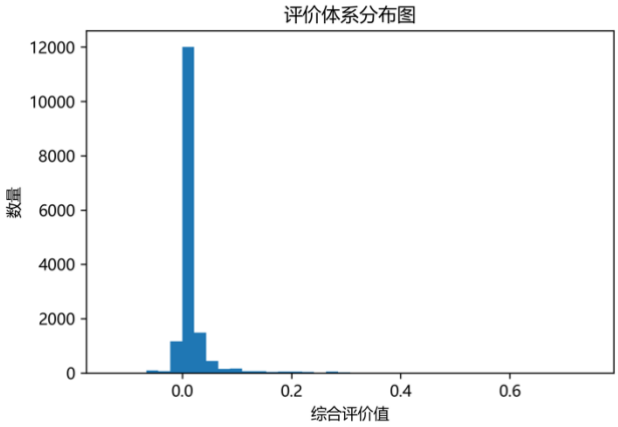

图 22 综合评价模型分布图

七、问题三模型的建立与求解

7.1 基于随机森林的货运线路总成本与定价预测模型

为了能够预测货运线路的总价格与指导价格的定价，我们采用随机森林算法分别对于货运线路的总价格与指导价格建立预测模型。

7.1.1 货运线路总成本定价预测模型

在货运线路总成本定价预测模型中，将总里程、业务类型、需求类型 1、需求类型 2、是否续签、车辆长度、标的展示策略、打包类型、运输等级、地区类型、需求紧急程度、承包商和个体司机作为模型输入变量，将线路总成本作为输出变量，建立随机森林模型预测模型，并采用 5 折法，把数据集分为 5 份，按照 4：1 进行训练集和测试集的分配，并通过交叉验证的方式，进行模型预测效果的检验。

仿真实验表明，此预测模型对于货运线路总价格定价预测准确率达到 0.9926。

7.1.2 货运线路定价预测模型

在货运线路指导价定价预测模型中，将总里程、业务类型、需求类型 1、需求类型 2、是否续签、车辆长度、标的展示策略、打包类型、运输等级、地区类型、需求紧急程度、承包商、个体司机作为输入变量，将线路价格（不含税）作为输出变量，建立随机森林模型预测模型，并采用 5 折法，把数据集分为 5 份，按照 4：1 进行训练集和测试集的分配，并通过交叉验证的方式，进行模型预测效果的检验。

仿真实验表明，此预测模型对于货运线路指导价定价预测准确率达到 0.9836。

7.1.3 定价预测模型与原定价模式效果比较

为了比较定价预测模型与原有定价模式之间优劣，我们使用问题二中设计的溢价率指标和成本时效综合指标进行分析。图 23-26 给出了原有平台定价和我们设计的随机森林预测模型之间的比较，从图 23 和图 24 看，原有策略的溢价率分布分散，平均值为 0.3 而本模型的溢价率非常低，集中在 0 附近，平均约为 0.1，远低于原来的策略。从图 25 和图 26 看，原有策略的成本时效值分布在 0 的两侧，而我们设计的模型成本时效值均大于 0，模型的指标平均值比原来策略模型大 0.05 具有显著的优势。

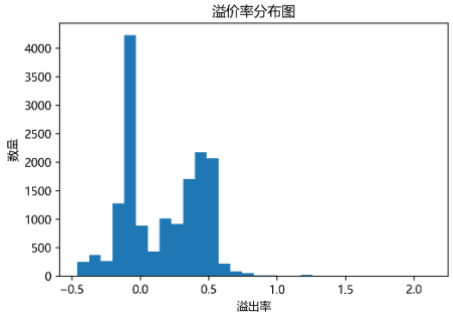

图 23 平台定价的溢价率分布图

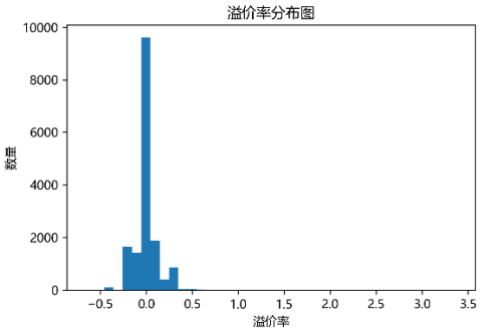

图 24 随机森林预测模型定价的溢价率分布图

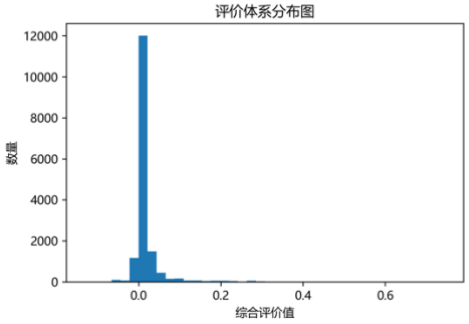

图 25 平台定价的综合评定分布图

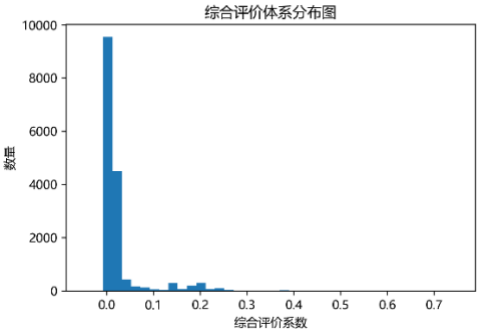

图 26 随机森林预测模型定价的综合评定分布图

我们很容易发现在评价体系与溢价率两方面，基于随机森林的货运线路总价格与指导价定价预测模型明显优于目前平台使用的定价模型。

7.2 动态定价模型的建立

由上文得知，基于随机森林的货运线路总价格与指导价定价预测模型能够给出极为准确的货运线路总价格定价与指导价定价预测。为了能够降低成本，且尽量保障最后成交，我们需要研究如何通过动态定价使得成本更低，且尽量不出现不成交的情况。因此需要研究承包商、个体司机第一次调价、第二次调价与第三次调价的规律。

首先，通过数据筛选，发现仅有交易反馈次数，即承包商、个体司机议价反馈次数，拥有时间轴。这就意味着承包商、个体司机议价反馈次数可能与调价比率有关。

为了验证承包商、个体司机议价反馈次数是否与调价比率有关，分别绘制出承包商、个体司机的议价反馈次数的箱形图见图 27 和 28：

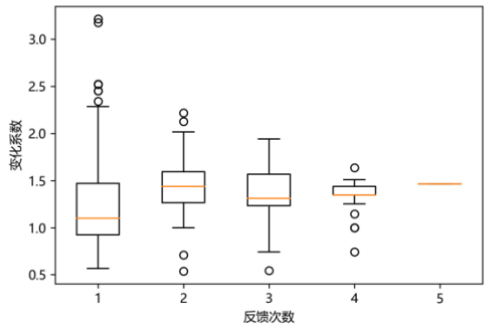

图 27 承包商议价反馈次数箱形图

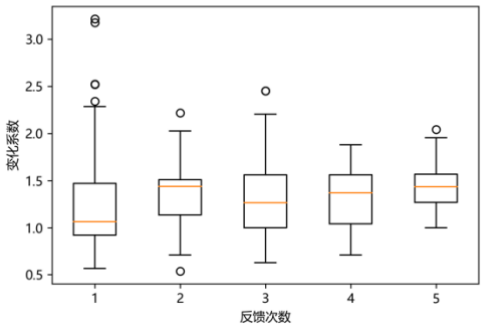

图 28 个体司机议价反馈次数箱形图

绘制出二者的中位数预测曲线则能够发现二者中位数曲线都为先上升后趋平的对数形式。故采用带有偏移值的对数函数回归进行拟合，建立动态定价模型。拟合情况

如图 29 和 30 所示：

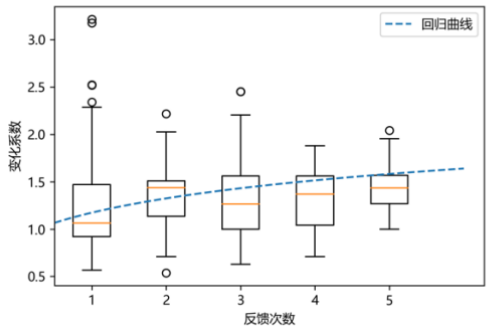

图 29 承包商议价反馈次数调整后曲线

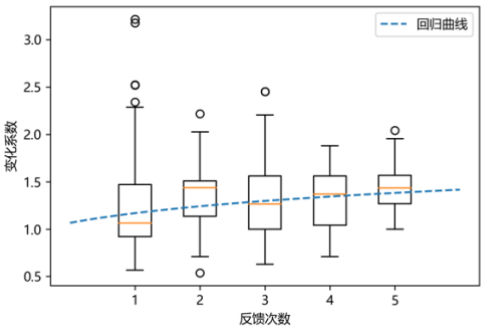

图 30 个体司机议价反馈次数调整后曲线

因此，得到承包商、个体司机第一次调价、第二次调价与第三次调价的预测比例分别为：

承包商第一次、第二次与第三次调价的比例分别为 0.9150、1.287、1.659；

个体司机第一次、第二次与第三次调价的比例分别为 0.8906、1.143、1.395。

根据交易对象的不同，如果承包商、个体司机分别议价，则直接选择议价段所对应的比例，如果承包商、个体司机同时议价，择取比例的平均值；

由此动态定价模型可知，基于随机森林的货运线路总价格与指导价定价预测模型与平台现有定价模型的预测调价比率较为一致，即承包商、个体司机不管对与交易的结果是否满意，依旧会进行议价行为。因此，交易反馈次数依旧存在，故认为调价比率几乎不变，即定价足够稳定。

八、结论分析——对于无车承运人平台运营建议信

无车承运人平台负责人：

你好！

随着我国无车承运行业逐步兴起，为满足贵无车承运人平台快速促进成交和降低承运成本，在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)的承运端司机的条件下，我们建议：

对于线路的定价，通过我们建立的多元线性回归模型以及随机森林模型，我们认为总里程、车辆长度、业务类型、运输等级、需求紧急程度以及打包类型是贵平台可参考的定价因素。此外，基于我们对货运运输总路程会对定价方法有一定影响的发现，建议对于总里程超过 2000 公里的长距离运输，如跨省货运定价，贵平台可着重参考该订单是否续签、需求紧急程度以及打包类型来拟定价格；而对于总里程少于 500 公里的短距离运输，如同一省市内的货运定价，贵平台可着重参考订单的总里程，结合承接对应订单的车辆的长度进行定价。

对于贵平台的历史订单，我们取用了溢价率作为判定定价效果的参考，综合评估了贵平台定价模型对于无车承运人的风险大小，发现贵平台目前的定价模型存在一定的风险。通过对比各因素在采取定价效果分类前后情况，发现对于订单定价效果，业务类型产生的影响非常大，而表的展示策略和是否续签情况也是产生了一定程度的影响。对应的，我们建议贵平台可以参考订单的业务类型，表的展示策略和是否续签情况以制定定价方案。

同时根据我们建立的对于定价的综合评价指标模型发现，贵平台历史订单的定价相对比较合理，从定价效果来看已经是一个相对合理的模型。

为了进一步更有效地提高定价预测准确度，减少后期由于承运商、个体司机端议价情况的频繁产生而导致的多次的订单价格调整，我们建立了基于随机森林的预测模型。其分别对总成本预测准确度达到 99.3%，定价预测准确度达到 98%。我们建议，贵平台也可参考该模型建立对订单拟定的价格更为准确的定价模型。

对于初次定价之后的调价策略，我们通过对历年数据进行的动态定价分析，发现调价比率与交易的反馈次数相关。为在保证每个任务必须被承运的前提下尽量降低承运成本，依据承运商议价反馈数量，第一次、第二次以及第三次调价的调价预测比例分别为 91.50%、128.7%与 165.9%为相对最优；而依据个体司机议价反馈数量，第一次、第二次以及第三次调价的调价预测比例分别为 89.06%、114.3%与 139.5%为相对最优。因为数据量受限，议价信息的缺失，我们得到的这一结论仅仅基于所给数据集上，而如果贵平台在实际调价中拥有更为翔实的议价信息的数据，在我们的基础上进一步结合更加详细的议价信息，那么贵平台的调价策略效果会更佳。

MathorCup 参赛选手

2020/5/25

九、模型优缺点与改进方向

9.1 模型优点

1. 变量的清理与选取避免了维度过高而导致的过拟合与维度诅咒；
2. 多元线性回归模型提供的回归分析可以准确地计量各个因素之间的相关程度与回归拟合程度的高低，提高预测方程式的效果；
3. 随机森林模型具有可以有效地在大数据集上运行、具有极高的准确率、能对于缺省值问题获得良好的结果、能够避免内部生成误差等优点；
4. 介于一个属性的信息增益越大，表明属性对样本的熵减少的能力越强，这个属性使得数据由不确定性变成确定性的能力越强，信息增益的使用提供了影响无车承运人平台进行货运的线路定价主要因素更加强有力的佐证；
5. 溢价率模型中加入了时间元素为定价评估提供了更为量化的定价评价标准；
6. 根据反馈数量进行模型的调整预测，解决了部分在没有实时数据情况下的时序性预测问题。

9.2 模型缺点与改进方向

1. 在回归模型以及随机森林模型中缺少连续变量，模型预测可能会发生轻微偏移；
2. 主要依靠回归模型以及随机森林模型。采用的模型类型较为单一，可添加其他的模型如 xgboost，adamboost，神经网络算法等进行佐证观点与结论；
3. 在经过清理的数据散点图中可以清晰地看到，长距离运输的相关数据的离散程度相比于短距离运输较大。对于这种具有较为明显离散的数据集的预测可能需要进行分类型的预测模型训练；
4. 在设计的综合评价指标模型中，所采用的因变量为需求紧急程度。不确定是否存在其他变量为因变量时能更加准确且有效地提供评估指标；
5. 如果拥有更为详细的每次反馈的调价、调价的具体时间、调价幅度大小等关键议价数据，则动态定价模型可以拥有更大的提高空间，进而提高调价策略的效果。

十、参考文献

【1】 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(01):142-146

【2】 李玲，刘华文，徐晓丹,等. 基于信息增益的多标签特征选择算法[J]. 计算机科学，2015(07):58-62.

【3】 张茂胜. 基于B-S定价模型的权证风险度量研究[D]. 大连理工大学.

十一、 附录

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import MinMaxScaler
from matplotlib.font_manager import FontProperties
from sklearn.ensemble import RandomForestClassifier
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from xgboost.sklearn import XGBClassifier
from scipy import stats


# In[6]:


#  研究数据集中哪些变量属于虚拟变量或者分类变量
data = pd.read_csv("附件一.csv")
result = dict()
for name in data.columns:
    process = list(set(data[name]))
    if len(process)<20:
        result[name] = process
print("以下是列表中的分类变量:")
for i in result.keys():
    print(i,result[i])


# ##  问题一:重点因素分析

# In[402]:
```

```
data = pd.read_csv("附件一.csv")
# 任务 id 不变
# 总里程不变
# 业务类型 ['重货', '速运'] --> 重货：0，速运：1
business_type = {"重货":0,"速运":1}
data["业务类型"] = data["业务类型"].map(business_type)
# 需求类型 1 ['普通', '区域发运'] --> 普通:0，区域发运:1
need_type_1 = {"普通":0,"区域发运":1}
data["需求类型 1"] = data["需求类型 1"].map(need_type_1)
# 需求类型 2 ['临时', '计划'] --> 临时:0，计划:1
need_type_2 = {"临时":0,"计划":1}
data["需求类型 2"] = data["需求类型 2"].map(need_type_2)
# 线路价格（不含税）不变:y
# 路线编号？（不确定）
# 调价比例 不变:y
# 路线指导价（不含税）不变:y
# 是否续签 ['已分拨续签', '未知', '非续签', '续签 ECP 审批驳回或撤销'] --> 非续签:0,已分
拨续签:1,续签 ECP 审批驳回或撤销:2,未知:nan
xuqian = {"非续签":0,"已分拨续签":1,"续签 ECP 审批驳回或撤销":2}
data["是否续签"] = data["是否续签"].map(xuqian)
# 始发网点:不变？去除？
del data["始发网点"]
# 实际到车时间：去除？
del data["实际到车时间"]
# 目的网点：不变？去除？
del data["目的网点"]
# 交易成功时长：不变？
# 交易成功时间：去除？
del data["交易成功时间"]
# 交易成功日期：去除？
del data["交易成功日期"]
# 计划发车时间：去除？
del data["计划发车时间"]
# 计划发车日期：去除？
del data["计划发车日期"]
# 分拨时间：去除？
del data["分拨时间"]
# 成交对象 ['C', 'B']--> "B":0,"C":1
finish_target = {"B":0,"C":1}
data["成交对象"] = data["成交对象"].map(finish_target)
# 车辆长度与车辆吨位成正比（见下面一个 cell），所以保留一个
del data["车辆吨位"]
# 标的展示策略 ['DIR', 'BDC'] --> DIR:0，BDC:1
show_type = {"DIR":0,"BDC":1}
```

```
data["标的展示策略"] = data["标的展示策略"].map(show_type)
# 打包类型 ['单边', '周期流向', '人工', 'N', '周期往返'] --> 人工:0,单边:1,周期流向:2,周期往返:3,"N":nan
pack_type = {"人工":0,"单边":1,"周期流向":2,"周期往返":3}
data["打包类型"] = data["打包类型"].map(pack_type)
# C 端议价反馈数量不变：y?
# 最后一次询价时间：去除?
del data["最后一次询价时间"]
# 子包号：去除?
del data["子包号"]
# 装卸的次数，装的次数,卸的次数？缺失值过多，建议去除?
del data["装卸的次数"]
del data["装的次数"]
del data["卸的次数"]
# 运输等级 ['二级运输', '一级运输', '三级运输']--> '一级运输':0,'二级运输':1,'三级运输':2
transport_level = {'一级运输':0,'二级运输':1,'三级运输':2}
data["运输等级"] = data["运输等级"].map(transport_level)
# 调价审核完成时间：去除?
del data["调价审核完成时间"]
# 调价类型 ['未调整', '调低', '调高'] --> '调低'：0，'未调整'：1，'调高'：2 ： y?
changing_price = {'调低':0,'未调整':1,'调高':2}
data["调价类型"] = data["调价类型"].map(changing_price)
# 调价紧急程度 ['非常紧急', '常规', '紧急', 'N'] --> '常规':0,'紧急':1,'非常紧急':2,'N':nan
changing_emergency = {'常规':0,'紧急':1,'非常紧急':2}
data["调价紧急程度"] = data["调价紧急程度"].map(changing_emergency)
# 调价 ECP 创建时间 去除?
del data["调价 ECP 创建时间"]
# 是否需要装卸 去除
del data["是否需要装卸"]
# 始发地省份名称['陕西省', 'N', '广东省', '北京市', '河南省'] --> '陕西省':0,'广东省':1,'北京市':2,'河南省':3,"N":nan
start_area = {'陕西省':0,'广东省':1,'北京市':2,'河南省':3}
data["始发地省份名称"] = data["始发地省份名称"].map(start_area)
# 实际靠车时间 去除?
del data["实际靠车时间"]
# 实际结束时间 去除?
del data["实际结束时间"]
# 实际发车时间 去除?
del data["实际发车时间"]
# 任务状态 去除（全部为已完成）
del data["任务状态"]
# 车辆类型分类 去除（全部为厢式运输车）
del data["车辆类型分类"]
# 目的地省份名称 ['N', '广东省', '北京市', '河南省', '新疆维吾尔自治区']--> '北京市':0,'广东
```

```
省':1,'河南省':2,'新疆维吾尔自治区':3,"N":nan
#target_area = {'北京市':0,'广东省':1,'河南省':2,'新疆维吾尔自治区':3}
#data["目的地省份名称"]=data["目的地省份名称"].map(target_area)
del data["目的地省份名称"]
# 交易开始时间 去除
del data["交易开始时间"]
# 交易结束时间 去除
del data["交易结束时间"]
# 交易对象 拆分 B 和 C:  利用 onehot 编码方式
b = list()
c = list()
for i in data["交易对象"]:
    if i == "B":
        b.append(1)
        c.append(0)
    elif i == "C":
        b.append(0)
        c.append(1)
    else:
        b.append(1)
        c.append(1)
data["交易对象 B"]=b
data["交易对象 C"]=c
del data["交易对象"]
# 计划卸货完成时间，缺失值过多，不适用，删去
del data["计划卸货完成时间"]
# 计划卸货等待时长 不确定？大概是删去？
del data["计划卸货等待时长"]
# 计划靠车时间 删去
del data["计划靠车时间"]
# 计划到达时间 删去
del data["计划到达时间"]
# 地区类型 ['分拨区', '业务区'] --> "分拨区":0,"业务区":1
area_type = {"分拨区":0,"业务区":1}
data["地区类型"]=data["地区类型"].map(area_type)
# 地区,全部减一
data["地区"]=data["地区"]-1
# 标的状态 去除（全部为已经抢标）
del data["标的状态"]
# 标的_创建时间 去除
del data["标的_创建时间"]
# 标的_创建日期 去除
del data["标的_创建日期"]
# 异常状态 去除（全部为正常）
```

```
del data["异常状态"]
# 交易模式 ['抢单', '8', 'N', '14'] --> "抢单":0,"8":1,"14":2,"N":nan
deal_type = {"抢单":0,"8":1,"14":2}
data["交易模式"] = data["交易模式"].map(deal_type)
# 线路编码 删去
del data["线路编码"]
# C 端议价最低价：保留
# B 端议价最低价：保留
# B 端议价反馈数量 不变？：y?
# 线路总成本 不变：y?
# 需求状态 删去
del data["需求状态"]
# 需求紧急程度 ['常规订单', '紧急订单', '特急订单'] --> '常规订单':0,'紧急订单':1,'特急订单':2
need_emergency = {'常规订单':0,'紧急订单':1,'特急订单':2}
data["需求紧急程度"]=data["需求紧急程度"].map(need_emergency)
display(data.head())
data = data.set_index("任务 id")
data.to_csv("processed_data.csv")


# In[484]:


# 相关性例子
data = pd.read_csv("附件一.csv")
plt.plot(data["车辆长度"],data["车辆吨位"],".",label="数据")
reg = LinearRegression()
reg.fit(np.array(data["车辆长度"]).reshape(-1,1),data["车辆吨位"])
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.plot(np.linspace(np.min(data["车辆长度"]),np.max(data["车辆长度"]),1000),
         np.linspace(np.min(data["车辆长度"]),np.max(data["车辆长度"]),1000)*reg.coef_[0]+reg.intercept_,label="回归曲线")
plt.xlabel("车辆长度")
plt.ylabel("车辆吨位")
plt.legend()
plt.savefig("车辆吨位与车辆长度关系图.png",dpi=300)
plt.show()


# In[485]:


# 详细的数据关系图
```

```
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
plt.figure(figsize=(12,8))
sns.heatmap(abs(processed_data.corr()))
plt.title("数据相关性系数表")
plt.savefig("数据相关性系数表.png",dpi=300)
plt.show()


# 先用线性回归方法进行线路指导价预测

# In[405]:


import statsmodels.formula.api as sm
import statsmodels.api as sm
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
                "是否续签","车辆长度",
                "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
processed_data_x=sm.add_constant(processed_data_x)
y = list()
for i in processed_data_y:
    y.append(float(i))
reg = sm.OLS(np.array(y),processed_data_x)
clf = reg.fit()
feature_importance = sorted(list(zip(abs(clf.pvalues[1:]),feature[:-1])),reverse=True)
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
```

```
plt.xlabel("Pvalue")
plt.title("线性回归 pvalue 值")
plt.tight_layout()
plt.savefig("各变量线性回归后的 pvalue.png",dpi=300)
plt.show()


# In[397]:


import statsmodels.formula.api as sm
import statsmodels.api as sm
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
                "是否续签","车辆长度","标的展示策略",
                "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
processed_data_x=sm.add_constant(processed_data_x)
y = list()
for i in processed_data_y:
    y.append(float(i))
reg = sm.OLS(np.array(y),processed_data_x)
clf = reg.fit()
feature_importance = sorted(list(zip(abs(clf.params[1:]),feature[:-1])))
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("Coefficient")
plt.title("线性回归系数")
plt.tight_layout()
```

```
plt.savefig("各变量线性回归后的 coefficient.png",dpi=300)
plt.show()


# In[400]:


# 由于总里程占的相关性太强了，导致其余数据相关性不明显，因此将总里程这个变量去除，去关注其他的变量对于这模型的影响
import statsmodels.formula.api as sm
import statsmodels.api as sm
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["业务类型","需求类型 1","需求类型 2",
                "是否续签","车辆长度","标的展示策略",
                "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
processed_data_x=sm.add_constant(processed_data_x)
y = list()
for i in processed_data_y:
    y.append(float(i))
reg = sm.OLS(np.array(y),processed_data_x)
clf = reg.fit()
feature_importance = sorted(list(zip(abs(clf.params[1:]),feature[:-1])))
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("Coefficient")
plt.title("线性回归系数（无总里程）")
plt.tight_layout()
plt.savefig("各变量线性回归后的 coefficient--无总里程.png",dpi=300)
```

```
plt.show()


# In[349]:


# 由于总里程占的相关性太强了，导致其余数据相关性不明显，因此将总里程这个变量去除，
去关注其他的变量对于这模型的影响
import statsmodels.formula.api as sm
import statsmodels.api as sm
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["业务类型","需求类型 1","需求类型 2",
                "是否续签","车辆长度","标的展示策略",
                "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
processed_data_x=sm.add_constant(processed_data_x)
y = list()
for i in processed_data_y:
    y.append(float(i))
reg = sm.OLS(np.array(y),processed_data_x)
clf = reg.fit()
feature_importance = sorted(list(zip(abs(clf.pvalues[1:]),feature[:-1])),reverse=True)
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("pvalue")
plt.title("pvalue for OLS without distance")
plt.savefig("各变量线性回归后的 pvalue--无总里程.png")
plt.show()
```

In[409]:

```
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
                "是否续签","车辆长度","标的展示策略",
                "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
y = list()
for i in processed_data_y:
    y.append(float(i))
print("The cross validation accruacy is",np.mean(score))
rf.fit(processed_data_x,processed_data_y)
print("The feature importance based on the feature importance is ")
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
process = pd.DataFrame()
feature_importance = sorted(list(zip(rf.feature_importances_,feature[:-1])))
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("feature importance")
plt.title("随机森林的特征重要性")
plt.tight_layout()
plt.savefig("随机森林的 feature importance.png",dpi=300)
plt.show()
#print(feature_importance)
```

In[403]:

```
plt.plot(data["总里程"],data["线路指导价（不含税）"],".")
plt.plot([500 for i in range(2)],np.linspace(np.min(data["线路指导价（不含税）"]),
                                             np.max(data["线路指导价（不含税）
"])
                                            ,2,".."),label="500 公里分界线")
plt.plot([2000 for i in range(2)],np.linspace(np.min(data["线路指导价（不含税）"]),
                                             np.max(data["线路指导价（不含税）
"])
                                            ,2,"-."),label="2000 公里分界线")
plt.xlabel("总里程数")
plt.ylabel("指导价")
plt.title("总里程与路线指导价的关系图")
plt.legend()
plt.savefig("总里程与路线指导价的关系图.png",dpi=300)
plt.show()


# In[495]:


processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
           "是否续签","车辆长度","标的展示策略",
           "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["总里程"]>500]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
# 建立 random forest
rf = RandomForestRegressor(n_estimators=1000,max_depth=4)
score = cross_val_score(rf,processed_data_x,processed_data_y,cv=5)
print("The cross validation accruacy is",np.mean(score))
rf.fit(processed_data_x,processed_data_y)
print("The feature importance based on the feature importance is ")
```

```
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
process = pd.DataFrame()
feature_importance = sorted(list(zip(rf.feature_importances_,feature[:-1])))
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("feature importance")
plt.title("长距离的特征重要性")
plt.tight_layout()
plt.savefig("长距离的 feature importance.png",dpi=300)
plt.show()
#print(feature_importance)


# In[494]:


processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
           "是否续签","车辆长度","标的展示策略",
           "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["总里程"]<500]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
# 建立 random forest
rf = RandomForestRegressor(n_estimators=1000,max_depth=4)
score = cross_val_score(rf,processed_data_x,processed_data_y,cv=5)
print("The cross validation accruacy is",np.mean(score))
rf.fit(processed_data_x,processed_data_y)
print("The feature importance based on the feature importance is ")
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
process = pd.DataFrame()
```

```
feature_importance = sorted(list(zip(rf.feature_importances_,feature[:-1])))
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("feature importance")
plt.title("短距离的特征重要性")
plt.tight_layout()
plt.savefig("短距离的 feature importance.png",dpi=300)
plt.show()
#print(feature_importance)


# In[351]:


import statsmodels.formula.api as sm
import statsmodels.api as sm
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
           "是否续签","车辆长度","标的展示策略",
           "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x[processed_data_x["总里程"]>500]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
processed_data_x=sm.add_constant(processed_data_x)
y = list()
for i in processed_data_y:
    y.append(float(i))
reg = sm.OLS(np.array(y),processed_data_x)
clf = reg.fit()
feature_importance = sorted(list(zip(abs(clf.pvalues[1:]),feature[:-1])),reverse=True)
```

```
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("Pvalue")
plt.title("Pvalue for OLS for long distance")
plt.savefig("各变量线性回归后的 pvalue--more than 500km.png")
plt.show()


# In[350]:


import statsmodels.formula.api as sm
import statsmodels.api as sm
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
           "是否续签","车辆长度","标的展示策略",
           "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x[processed_data_x["总里程"]<500]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
processed_data_x=sm.add_constant(processed_data_x)
y = list()
for i in processed_data_y:
    y.append(float(i))
reg = sm.OLS(np.array(y),processed_data_x)
clf = reg.fit()
feature_importance = sorted(list(zip(abs(clf.pvalues[1:]),feature[:-1])),reverse=True)
importance = list()
name = list()
for i in feature_importance:
```

```
        importance.append(i[0])
        name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("Pvalue")
plt.title("Pvalue for OLS for short distance")
plt.savefig("各变量线性回归后的 pvalue--less than 500km.png")
plt.show()


# In[487]:


import statsmodels.formula.api as sm
import statsmodels.api as sm
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
                "是否续签","车辆长度","标的展示策略",
                "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x[processed_data_x["总里程"]>500]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
processed_data_x=sm.add_constant(processed_data_x)
y = list()
for i in processed_data_y:
        y.append(float(i))
reg = sm.OLS(np.array(y),processed_data_x)
clf = reg.fit()
feature_importance = sorted(list(zip(abs(clf.params[1:]),feature[:-1])))
importance = list()
name = list()
for i in feature_importance:
        importance.append(i[0])
        name.append(i[1])
plt.barh(y=name,width=importance)
```

```
plt.xlabel("Coefficient")
plt.title("长路程的线性回归")
plt.savefig("各变量线性回归后的 coefficient -- more than 500.png",dpi=300)
plt.show()


# In[489]:


import statsmodels.formula.api as sm
import statsmodels.api as sm
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
                "是否续签","车辆长度","标的展示策略",
                "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x[processed_data_x["总里程"]<500]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
processed_data_x=sm.add_constant(processed_data_x)
y = list()
for i in processed_data_y:
    y.append(float(i))
reg = sm.OLS(np.array(y),processed_data_x)
clf = reg.fit()
feature_importance = sorted(list(zip(abs(clf.params[1:]),feature[:-1])))
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("Coefficient")
plt.title("短路程的线型回归")
plt.savefig("各变量线性回归后的 coefficient -- less than 500.png",dpi=300)
```

```
plt.show()


# In[497]:


# 信息熵与信息增益
def calc_ent(x):
    """
        calculate shanno ent of x
    """

    x_value_list = set([x[i] for i in range(x.shape[0])])
    ent = 0.0
    for x_value in x_value_list:
        p = float(x[x == x_value].shape[0]) / x.shape[0]
        logp = np.log2(p)
        ent -= p * logp

    return ent
def calc_condition_ent(x, y):
    """
        calculate ent H(y|x)
    """

    # calc ent(y|x)
    x_value_list = set([x[i] for i in range(x.shape[0])])
    ent = 0.0
    for x_value in x_value_list:
        sub_y = y[x == x_value]
        temp_ent = calc_ent(sub_y)
        ent += (float(sub_y.shape[0]) / y.shape[0]) * temp_ent

    return ent
def calc_ent_grap(x,y):
    """
        calculate ent grap
    """

    base_ent = calc_ent(y)
    condition_ent = calc_condition_ent(x, y)
    ent_grap = base_ent - condition_ent

    return ent_grap
```

```
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
                "是否续签","车辆长度","标的展示策略",
                "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路指导价（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路指导价（不含税）"]!="N"]
processed_data_x = processed_data_x[processed_data_x["总里程"]<500]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路指导价（不含税）"]
# 删除 output 量
del processed_data_x["线路指导价（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
result = list()
for i in range(13):
    result.append(calc_ent_grap(processed_data_x[:,i],np.array(processed_data_y)))
result = np.array(result)
feature_importance = sorted(list(zip(abs(result),feature[:-1])))
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)

plt.title("信息增益")
plt.tight_layout()
plt.savefig("各变量的信息增益.png",dpi=300)
plt.show()


# ## 第二问

# In[490]:


data = pd.read_csv("processed_data.csv")
data = data[data["线路总成本"]!="N"]
#display(data["线路总成本"])
chenben = list()
```

```
for i in data["线路总成本"]:
    chenben.append(float(i))
#display(data["线路价格（不含税）"])
result = (data["线路价格（不含税）"]-np.array(chenben))/np.array(chenben)
plt.hist(result,bins=30)
plt.xlabel("溢价率")
plt.ylabel("数量")
plt.title("溢价率分布图")
plt.savefig("溢价率分布图.png",dpi=300)
plt.show()


# In[377]:


processed_data = pd.read_csv("processed_data.csv")
processed_data = processed_data[processed_data["线路总成本"]!="N"]
#display(processed_data.head())
#plt.hist(processed_data["总里程"])
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
           "是否续签","车辆长度","标的展示策略",
           "打包类型","运输等级","地区类型","需求紧急程度"]
for name in feature:
    plt.subplot(1,2,1)
    plt.hist(processed_data[name])
    plt.title(name)
    plt.ylabel("number")
    plt.subplot(1,2,2)
    plt.hist(processed_data[result>0.4][name])
    plt.title(name+"-溢出高")
    plt.savefig(name+"-溢出高.png",dpi=300)
    plt.show()


# In[498]:


data = pd.read_csv("processed_data.csv")
mms = MinMaxScaler()
index = ["需求紧急程度","线路价格（不含税）","线路指导价（不含税）","交易成功时长"]
data = data[index]
data = data[data["线路指导价（不含税）"]!="N"]
data = pd.DataFrame(mms.fit_transform(data))
data.columns = index
```

```
data["需求紧急程度"]=(data["需求紧急程度"]+0.5)*0.5
result = (data["线路指导价（不含税）"]-data["线路价格（不含税）"])*(1-data["需求紧急程度"])+data["交易成功时长"]*data["需求紧急程度"]
plt.hist(result,bins=40)
plt.xlabel("综合评价值")
plt.ylabel("数量")
plt.title("评价体系分布图")
plt.savefig("评价体系分布.png",dpi=300)
plt.show()


# In[450]:


processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
# 选取特征
feature = ["总里程","业务类型",
                "车辆长度",
                "打包类型","运输等级","需求紧急程度","线路价格（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路价格（不含税）"]
# 删除 output 量
del processed_data_x["线路价格（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
# 建立 random forest
rf = RandomForestRegressor(n_estimators=1000,max_depth=4)
score = cross_val_score(rf,processed_data_x,processed_data_y,cv=5)
print("The cross validation accruacy is",np.mean(score))
rf.fit(processed_data_x,processed_data_y)
print("The feature importance based on the feature importance is ")
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
process = pd.DataFrame()
feature_importance = sorted(list(zip(rf.feature_importances_,feature[:-1])))
importance = list()
name = list()
for i in feature_importance:
    importance.append(i[0])
    name.append(i[1])
plt.barh(y=name,width=importance)
plt.xlabel("feature importance")
```

```
plt.title("feature importance for random forest for reality prediction")
#plt.savefig("长距离的 feature importance.png")
plt.show()
#print(feature_importance)


# In[451]:


prediction = rf.predict(processed_data_x)
processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
feature = ["总里程","业务类型",
                "车辆长度",
                "打包类型","运输等级","需求紧急程度","线路价格（不含税）"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x.dropna()
result = (processed_data_x["线路价格（不含税）"]-prediction)/prediction
plt.hist(result,bins=40)
plt.xlabel("溢价率")
plt.ylabel("数量")
plt.title("溢价率分布图")
plt.savefig("溢价率分布图--rf 预测.png",dpi=300)
plt.show()


# In[452]:


processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
feature = ["总里程","业务类型","需求类型 1","需求类型 2",
                "是否续签","车辆长度","标的展示策略",
                "打包类型","运输等级","地区类型","需求紧急程度","交易对象 B","交易对象 C","
线路价格（不含税）","交易成功时长"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x.dropna()
mms = MinMaxScaler()
processed_data_x["prediction"] = prediction
processed_data_x = pd.DataFrame(mms.fit_transform(processed_data_x))
processed_data_x.columns = feature+["prediction"]
prediction = processed_data_x["prediction"]
processed_data_x["需求紧急程度"]=(processed_data_x["需求紧急程度"]+0.5)*0.5
result = (prediction-processed_data_x["线路价格（不含税）"])*(1-processed_data_x["需求紧
急程度"])+processed_data_x["交易成功时长"]*processed_data_x["需求紧急程度"]
print(result)
```

```
plt.hist(result,bins=40)
plt.xlabel("综合评价系数")
plt.ylabel("数量")
plt.title("综合评价体系分布图")
plt.savefig("评价体系分布图.png",dpi=300)
plt.show()


# ##  问题三

# In[380]:


processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
#  选取特征
feature = ["总里程","业务类型",
                "车辆长度",
                "打包类型","运输等级","需求紧急程度","线路总成本"]
processed_data_x = processed_data[feature]
processed_data_x = processed_data_x[processed_data_x["线路总成本"]!="N"]
processed_data_x = processed_data_x.dropna()
#  提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路总成本"]
#  删除 output 量
del processed_data_x["线路总成本"]
#  数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
#  建立 random forest
rf1 = RandomForestRegressor(n_estimators=1000,max_depth=4)
score = cross_val_score(rf1,processed_data_x,processed_data_y,cv=5)
print("The cross validation accruacy is",np.mean(score))
rf1.fit(processed_data_x,processed_data_y)


# In[381]:


processed_data = pd.read_csv("processed_data.csv").set_index("任务 id")
#  选取特征
feature = ["总里程","业务类型",
                "车辆长度",
                "打包类型","运输等级","需求紧急程度","线路价格（不含税）"]
processed_data_x = processed_data[feature]
```

```
processed_data_x = processed_data_x[processed_data_x["线路价格（不含税）"]!="N"]
processed_data_x = processed_data_x.dropna()
# 提取输出的线路指导价作为 output
processed_data_y = processed_data_x["线路价格（不含税）"]
# 删除 output 量
del processed_data_x["线路价格（不含税）"]
# 数据归一化
mms = MinMaxScaler()
processed_data_x = mms.fit_transform(processed_data_x)
# 建立 random forest
rf2 = RandomForestRegressor(n_estimators=1000,max_depth=4)
score = cross_val_score(rf2,processed_data_x,processed_data_y,cv=5)
print("The cross validation accruacy is",np.mean(score))
rf2.fit(processed_data_x,processed_data_y)


# In[482]:


data = pd.read_csv("附件二.csv")
# 任务 id 不变
# 总里程不变
# 业务类型 ['重货', '速运'] --> 重货：0，速运：1
business_type = {"重货":0,"速运":1}
data["业务类型"] = data["业务类型"].map(business_type)
# 需求类型 1 ['普通', '区域发运'] --> 普通:0，区域发运:1
need_type_1 = {"普通":0,"区域发运":1}
data["需求类型 1"] = data["需求类型 1"].map(need_type_1)
# 需求类型 2 ['临时', '计划'] --> 临时:0，计划:1
need_type_2 = {"临时":0,"计划":1}
data["需求类型 2"] = data["需求类型 2"].map(need_type_2)
# 线路价格（不含税）不变:y
# 路线编号？（不确定）
# 调价比例 不变:y
# 路线指导价（不含税）不变:y
# 是否续签 ['已分拨续签', '未知', '非续签', '续签 ECP 审批驳回或撤销'] --> 非续签:0,已分
拨续签:1,续签 ECP 审批驳回或撤销:2,未知:nan
xuqian = {"非续签":0,"已分拨续签":1,"续签 ECP 审批驳回或撤销":2}
data["是否续签"] = data["是否续签"].map(xuqian)
# 始发网点:不变？去除？
del data["始发网点"]
# 目的网点：不变？去除？
del data["目的网点"]
# 交易成功时长：不变？
```

```
# 计划发车时间：去除？
del data["计划发车时间"]
# 计划发车日期：去除？
del data["计划发车日期"]
# 分拨时间：去除？
del data["分拨时间"]
# 车辆长度与车辆吨位成正比（见下面一个 cell），所以保留一个
del data["车辆吨位"]
# 标的展示策略 ['DIR', 'BDC'] --> DIR:0，BDC:1
show_type = {"DIR":0,"BDC":1}
data["标的展示策略"] = data["标的展示策略"].map(show_type)
# 打包类型 ['单边', '周期流向', '人工', 'N', '周期往返'] --> 人工:0,单边:1,周期流向:2,周期往返:3,"N":nan
pack_type = {"人工":0,"单边":1,"周期流向":2,"周期往返":3}
data["打包类型"] = data["打包类型"].map(pack_type)
# C 端议价反馈数量不变：y
# 子包号：去除？
del data["子包号"]
# 装卸的次数，装的次数,卸的次数？缺失值过多，建议去除？
del data["装卸的次数"]
del data["装的次数"]
del data["卸的次数"]
# 运输等级 ['二级运输', '一级运输', '三级运输']--> '一级运输':0,'二级运输':1,'三级运输':2
transport_level = {'一级运输':0,'二级运输':1,'三级运输':2}
data["运输等级"] = data["运输等级"].map(transport_level)
# 是否需要装卸 去除
del data["是否需要装卸"]
# 始发地省份名称['陕西省', 'N', '广东省', '北京市', '河南省'] --> '陕西省':0,'广东省':1,'北京市':2,'河南省':3,"N":nan
start_area = {'陕西省':0,'广东省':1,'北京市':2,'河南省':3}
data["始发地省份名称"] = data["始发地省份名称"].map(start_area)
# 车辆类型分类 去除（全部为厢式运输车）
del data["车辆类型分类"]
# 目的地省份名称 ['N', '广东省', '北京市', '河南省', '新疆维吾尔自治区']--> '北京市':0,'广东省':1,'河南省':2,'新疆维吾尔自治区':3,"N":nan
#target_area = {'北京市':0,'广东省':1,'河南省':2,'新疆维吾尔自治区':3}
#data["目的地省份名称"]=data["目的地省份名称"].map(target_area)
del data["目的地省份名称"]
# 交易对象 拆分 B 和 C:  利用 onehot 编码方式
b = list()
c = list()
for i in data["交易对象"]:
    if i == "B":
        b.append(1)
```

```
            c.append(0)
        elif i == "C":
            b.append(0)
            c.append(1)
        else:
            b.append(1)
            c.append(1)
data["交易对象 B"]=b
data["交易对象 C"]=c
del data["交易对象"]
# 计划卸货完成时间，缺失值过多，不适用，删去
del data["计划卸货完成时间"]
# 计划卸货等待时长 不确定？大概是删去？
del data["计划卸货等待时长"]
# 计划靠车时间 删去
del data["计划靠车时间"]
# 计划到达时间 删去
del data["计划到达时间"]
# 地区类型 ['分拨区', '业务区'] --> "分拨区":0,"业务区":1
area_type = {"分拨区":0,"业务区":1}
data["地区类型"]=data["地区类型"].map(area_type)
# 地区,全部减一
data["地区"]=data["地区"]-1
# 标的_创建时间 去除
del data["标的_创建时间"]
# 标的_创建日期 去除
del data["标的_创建日期"]
# 异常状态 去除（全部为正常）
del data["异常状态"]
# 交易模式 ['抢单', '8', 'N', '14'] --> "抢单":0,"8":1,"14":2,"N":nan
deal_type = {"抢单":0,"8":1,"14":2}
data["交易模式"] = data["交易模式"].map(deal_type)
# 线路编码 删去
del data["线路编码"]
# C 端议价最低价：保留
# B 端议价最低价：保留
# B 端议价反馈数量 不变？：y?
# 线路总成本 不变：y?
# 需求紧急程度 ['常规订单', '紧急订单', '特急订单'] --> '常规订单':0,'紧急订单':1,'特急订单
':2
need_emergency = {'常规订单':0,'紧急订单':1,'特急订单':2}
data["需求紧急程度"]=data["需求紧急程度"].map(need_emergency)
display(data.head())
data = data.set_index("任务 id")
```

```
data.to_csv("processed_predict_data.csv")


# In[385]:


processed_predict_data = pd.read_csv("processed_predict_data.csv")
feature = ["总里程","业务类型",
                "车辆长度",
                "打包类型","运输等级","需求紧急程度"]
fillnamap = dict(zip(feature,stats.mode(processed_predict_data[feature])[0][0]))
processed_predict_data = processed_predict_data.fillna(fillnamap)[feature]
#  数据归一化
mms = MinMaxScaler()
processed_predict_data = mms.fit_transform(processed_predict_data)
result_zongchengben = rf1.predict(processed_predict_data)


# In[386]:


processed_predict_data = pd.read_csv("processed_predict_data.csv")
feature = ["总里程","业务类型",
                "车辆长度",
                "打包类型","运输等级","需求紧急程度"]
fillnamap = dict(zip(feature,stats.mode(processed_predict_data[feature])[0][0]))
processed_predict_data = processed_predict_data.fillna(fillnamap)[feature]
#  数据归一化
mms = MinMaxScaler()
processed_predict_data = mms.fit_transform(processed_predict_data)
result_baojia = rf2.predict(processed_predict_data)


#  与交易的反馈次数有关

# In[454]:


processed_data_x = pd.read_csv("processed_data.csv").set_index("任务 id")
plt.plot(processed_data_x["B 端议价反馈数量"],processed_data_x["调价比例"],".")
plt.show()
process = [list(processed_data_x[processed_data_x["B 端议价反馈数量"]==i]["调价比例"]) for
i in range(5)]
plt.boxplot(process)
```

```
plt.show()


# In[477]:


log_feedback = np.log(np.array(processed_data_x["B 端议价反馈数量"]+2))
reg = LinearRegression()
reg.fit(np.array(log_feedback).reshape(-1,1),processed_data_x["调价比例"])
x = np.linspace(-0.5,5,1000)
y = reg.predict(np.log(x+2).reshape(-1,1))
plt.boxplot(process)
plt.plot(x+1,y,"--",label="回归曲线")
plt.xlabel("反馈次数")
plt.ylabel("变化系数")
plt.legend()
plt.savefig("B 端议价分析.png",dpi=300)
reg.predict([[0],[1],[2]])


# In[478]:


processed_data_x = pd.read_csv("processed_data.csv").set_index("任务 id")
feedback = processed_data_x["C 端议价反馈数量"]
feedback_plot = list()
for i in feedback:
    if i>5:
        feedback_plot.append(5)
    else:
        feedback_plot.append(i)
plt.plot(feedback_plot,processed_data_x["调价比例"],".")
plt.show()
print(feedback_plot!=1)
process = [list(processed_data_x[np.array(feedback_plot)==i]["调价比例"]) for i in range(5)]
plt.boxplot(process)
plt.show()


# In[479]:


log_feedback = np.log(np.array(feedback_plot)+3)
reg = LinearRegression()
```

```
reg.fit(np.array(log_feedback).reshape(-1,1),processed_data_x["调价比例"])
x = np.linspace(-1.0,5,1000)
y = reg.predict(np.log(x+3).reshape(-1,1))
plt.boxplot(process)
plt.plot(x+1,y,"--",label="回归曲线")
plt.xlabel("反馈次数")
plt.ylabel("变化系数")
plt.legend()
plt.savefig("C 端议价分析.png",dpi=300)
reg.predict([[0],[1],[2]])


# In[483]:


final_result = pd.DataFrame()
final_result["id"]= data["编号"]
final_result["总成本"]=result_zongchengben
final_result["指导价"]=result_baojia


# In[ ]:
```