队伍编号	746
赛道	A

二手车估价问题

摘要

随着我国的机动车数量不断增长，人均保有量也随之增加，机动车以“二手车”形式在流通环节的流通需求越来越大。二手车作为一种特殊的“电商商品”，因为其“一车一况”的特性比一般电商商品的交易要复杂得多，因此，我们需要根据市场交易的实际情况，来评估二手车的各个特征综合对交易价格的影响，进而评估二手车资产的价值。而且，在一个典型的二手车零售场景中，二手车平台一般通过互联网线上推广渠道和方式获取用户需求，门店通过“买手”从个人或其他渠道收购二手车，然后由门店定价师定价销售，如果定价太高滞销也会打折促销，甚至直接以较低的价格打包批发，直至商品最终卖出，这涉及到二手车的保值率问题。所以，对二手车合理的定价以及挖掘影响二手车成交周期的关键因素，对二手车的销售具有重要作用。

在本文中，我们的数据预处理阶段分为数据清洗和数据转换两部分，在数据清洗的环节中，我们做了去除异常值和填补缺失值等工作，以便得到更好的训练模型。在数据转换的环节中，我们对数据的格式进行了统一，并且适当的改进了数据的表现形式，便于后续的训练和预测工作。

针对问题一，我们通过对数据的分析，发现数据中存在类别型和数值型的两种特征，这启发我们使用决策树模型，考虑到数据量并不是很大，所以我们采用了基于决策树算法的梯度提升框架模型 lightGBM，并对数据进行五折交叉验证，进一步提高预测精度。

针对问题二，我们在已有较多二手车交易数据的情况下，建立了随机森林的回归模型，通过随机森林回归模型，我们得到影响二手车成交周期的关键因素，并且得到各个因素的影响权重。进而对定价进行相应改进，以缩短交易周期。

在问题二解决后，由于我们只分析了影响车辆成交周期的关键因素，而对没有成交的车辆没有进行分析，于是我们对影响车辆能否成交的关键因素进行了分析处理，以期通过对车辆价格等因素，进行相应的改进，来增加二手车的成交率。

关键词：二手车 估价模型 lightGBM 算法 影响因素 随机森林算法 多元回归算法

目录

一、 问题重述

1.1 问题背景

近年来，二手车交易市场发展速度越来越快，根据相关报告，2020 年，中国的二手车交易量为 1434.14 万辆，其中二手乘用车交易量占新乘用车销量的百分比，对比于其他国家的二手车交易成熟市场（美国约为 239%，日本约为 137%），却仅有 71%。表明中国的二手车交易市场仍有巨大的提升空间。并且，由于我国汽车行业逐渐由周期成长行业转变为周期行业，汽车产业从增量市场转变为存量市场，国内的汽车保有量不断增加。未来，伴随着中国汽车保有量的进一步增加、二手车交易制度的健全、政策不断完善以及二手车不对称信息问题的解决，未来二手车市场的增长速度还会逐步加快。

目前，国内的二手车交易渠道主要分为线下传统销售和线上销售两种模式，由于线上交易的方便与快捷，网上二手车交易正成为二手车交易的主要方式。但由于卖场中的车源质量经常参差不齐，以及二手车“一车一价”的特性，同时私人之间的交易价格往往也会有较大的差异，这导致了对于二手车资产价值并没有一个标准的评估方案，使得二手车交易平台很难对线上销售的的二手车给出一个适宜的估价，这既导致了企业利润的减少，也导致了客户满意度的降低。若我们能够根据已交易的二手车数据来进行机器学习，通过分析得到二手车的交易特征，就可以合理的预测二手车的交易价格，从而提高客户的购买满意度以及企业的利润。

由于基于决策树算法的 lightGBM 模型是基于偏差的算法，所以对于数据中的噪声非常敏感，这就要求我们在数据清洗环节进行更高要求的处理，包括对于空缺值的填补，特征数据的降维。但是给出的特征比较多，直接进行训练维度太大，所以分析出哪些特征起到的影响十分微弱是很关键的一步，所以我们采用了相似分析的方法排除了几个特征，这有利于我们对二手车交易特点的精确分析，并且防止了过拟合的产生。分析二手车资产的价值对于国内二手车市场的完善和国家出台相应政策具有重要的研究价值。

同时，在比较典型的的二手车零售场景中，往往采取离线商务模式进行销售，即 020 商业模式。通过在互联网等线上渠道获取用户的线索后，门店再通

过“买手”从个人或者其它的渠道收购二手车，然后由门店的定价师进行定价销售。

这种模式下，如果不能正确的定价，便会造成二手车的积存，导致对二手车采取打折促销或者以低价打包批发售出的方式。这就要求我们对影响二手车成交周期的关键因素进行挖掘，进而在不影响利润的情况下，采取相应的措施加快门店销售车辆的销售速度。

利用和随机森林算法，我们可以综合考虑各个特征影响因素的大小。在比较传统的多元线性回归算法中，虽然我们可以对数据进行很好的解释，具有很强的解释性，但前提是我们需要假设数据和交易周期是线性相关的；而对于随机森林算法，由于它是以自主采样的方法为基础，所以会生成多元非线性的回归模型。这样，我们便可以避免对所有数据进行线性假设，大大提高了准确度。这对门店销售的效率改进提供了重要的价值。

1.2 问题重述

已知给定的二手车交易样本数据，其中包括估价训练数据，估价验证数据，门店交易训练数据。并且在问题一中给出了模型的评测标准公式。本文根据二手车的交易数据，解决题目要求的以下三个问题：

问题一：基于给定的二手车交易样本数据，进行了数据预处理，选用了lightGBM模型，进行训练并预测了二手车的零售交易价格，并利用估价验证数据进行了模型评估。

问题二：在给定的门店交易训练数据中，利用随机森林和多元回归，挖掘出影响车辆的成交周期的关键因素，给出了提高销售速度的手段并给出适用条件和预期效果。

问题三：在问题二解决的基础上，挖掘影响车辆能否成交的关键因素，并给出适当的分析，使得厂商可以合理的选择相应车型满足客户需求。

二、模型假设和字符说明

2.1 模型假设

假设一：假设只有题目提供的特征影响二手车的交易价格，其它因素不造成影响；

假设二：假设二手车的交易价格是有效的，不存在退换的情况。

假设三：假设因变量 x 在所抽样本中具有变异性，而且随着样本量的增加，因变量 x 的方差趋近于一个非零常数。

假设四：假设已知数据和未知数据服从相同的规律，所取样本具有代表性，能够反应整体特征

2.2 字符说明

LightGBM 参数	具体意义
num_leaves	生成的每棵树上的叶数
max_depth	每棵树的最大深度
max_bin	影响树的深度
min_data_in_leaf	每个桶内最少的数据量
feature_fraction	指定每次迭代所需要的特征部分
bagging_fraction	指定每次迭代所需要的数据部分
bagging_freq	每迭代多少轮，进行一次 b_f
lambda_l1	正则化系数 1
lambda_l2	正则化系数 2
min_split_gain	描述树分裂的最小 gain

三、问题一数据的预处理和初步分析

3.1 原始数据集的介绍

题目所给的三万组数据中，对主要特征类等信息进行了脱敏，主要数据包括了二手车辆的基础信息，如车辆颜色，行驶里数；交易时间信息，如展销时间，成交时间；价格信息等，共包含 36 列的变量信息，其中还包括有 15 列的

匿名变量信息，没有给出具体的实际意义。原始的数据集中具体的字段名和含义说明如下表 3.1 所示。

表 3.1 原始数据及字段名及含义说明

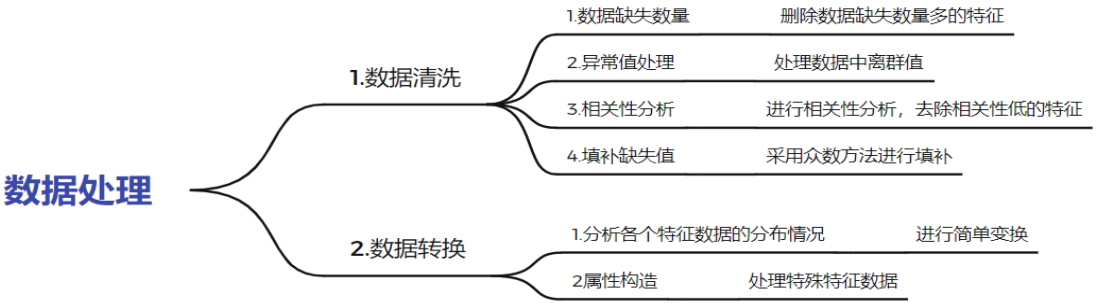
字段名	字段含义
TradeTime	交易车辆的展销时间
Brand	交易车辆的品牌 ID
Serial	交易车辆所属车系 ID
Model	交易车辆的车型 ID
Mileage	交易车辆的行驶里程
Color	交易车辆颜色
CityId	车辆所在城市 ID
Carcode	交易车辆国标码
Transfercount	交易车辆的过户次数
Seatings	交易车辆可载客人数
Registerdate	交易车辆注册日期
Licensedate	交易车辆上牌日期
Country	交易车辆的国别
Maketype	交易车辆的厂商 ID
Modelyear	交易车辆的年款
Displacement	交易车辆的排量
Gearbox	交易车辆的变速箱类型
Oiltype	交易车辆的燃油类型
Newprice	交易车辆的新车价格
Anonymousfeature（N）	15 个匿名特征（以 N 代替）
Price	交易车辆的交易价格

根据表 3.1，我们可以发现数据集中给出的特征比较多，采用传统的回归方法很容易产生过拟合的现象，导致虽然训练结果拟合程度高，但测验结果拟合程度差，并且训练时间十分的长。同时，我们亦可以发现，给出的数据中，既有数值型的特征信息也有类别类的特征信息，这就给我们启发采用决策树算

法，既可以解决离散型数据问题也可以解决连续性数值问题。因此本文综合考虑，采用了高效率且不易产生过拟合的 lightGBM 模型进行训练。

3.2 数据的预处理

对于数据的处理，我们总的数据处理思路为：



3.2.1 数据清洗

步骤一、首先我们原始数据集的数据进行分析，得到下图 3.1。

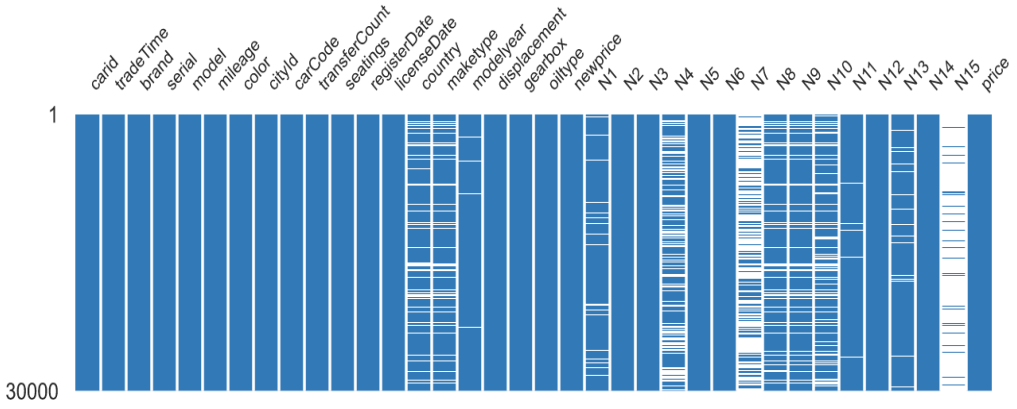


图 3.1

通过上图我们可以分析发现，只有少部分特征数据缺失量过多，在我们的综合考虑下，我们决定删除匿名特征 *N7* 与 *N15*。同时我们分析数据特征后发现，carid 这一特征并不是二手车交易的影响因素，所以我们删除了 carid 这一特征信息，

步骤二、处理数据中的异常值。我们对各个特征的数据进行了分析，其

中，在我们的分析下，我们发现在 country 和 price 特征中出现了较为显著的离

Value	Count	Frequency (%)
0	68	0.2%
779411	870	2.9%
779412	10054	33.5%
779413	3094	10.3%
779414	737	2.5%
779415	5809	19.4%
779416	3773	12.6%
779417	76	0.3%
779418	314	1.0%
779419	1174	3.9%

群值，而对于其他特征，我们没有发现显著的离群值，其中 country（图 3.2）和 price（图 3.3）的数据分析图如下：

图 3.2

Value	Count	Frequency (%)
109000	1	< 0.1%
678.8	1	< 0.1%
658	1	< 0.1%
596	1	< 0.1%
528.8	1	< 0.1%
459.8	1	< 0.1%
378	1	< 0.1%
358.8	1	< 0.1%
318	1	< 0.1%
296.8	1	< 0.1%

图 3.3

根据图中的数据我们可以很明显的看出，在 country 这一特征中，出现了错误值 0，对此我们采用众数进行填补，在 price 这一特征中，出现了极端的高成交价格用户，由于其数量十分少，所以我们对这组数据进行了删除处理。

步骤三、进行特征的相关性分析，作出类别特征交叉图 3.4

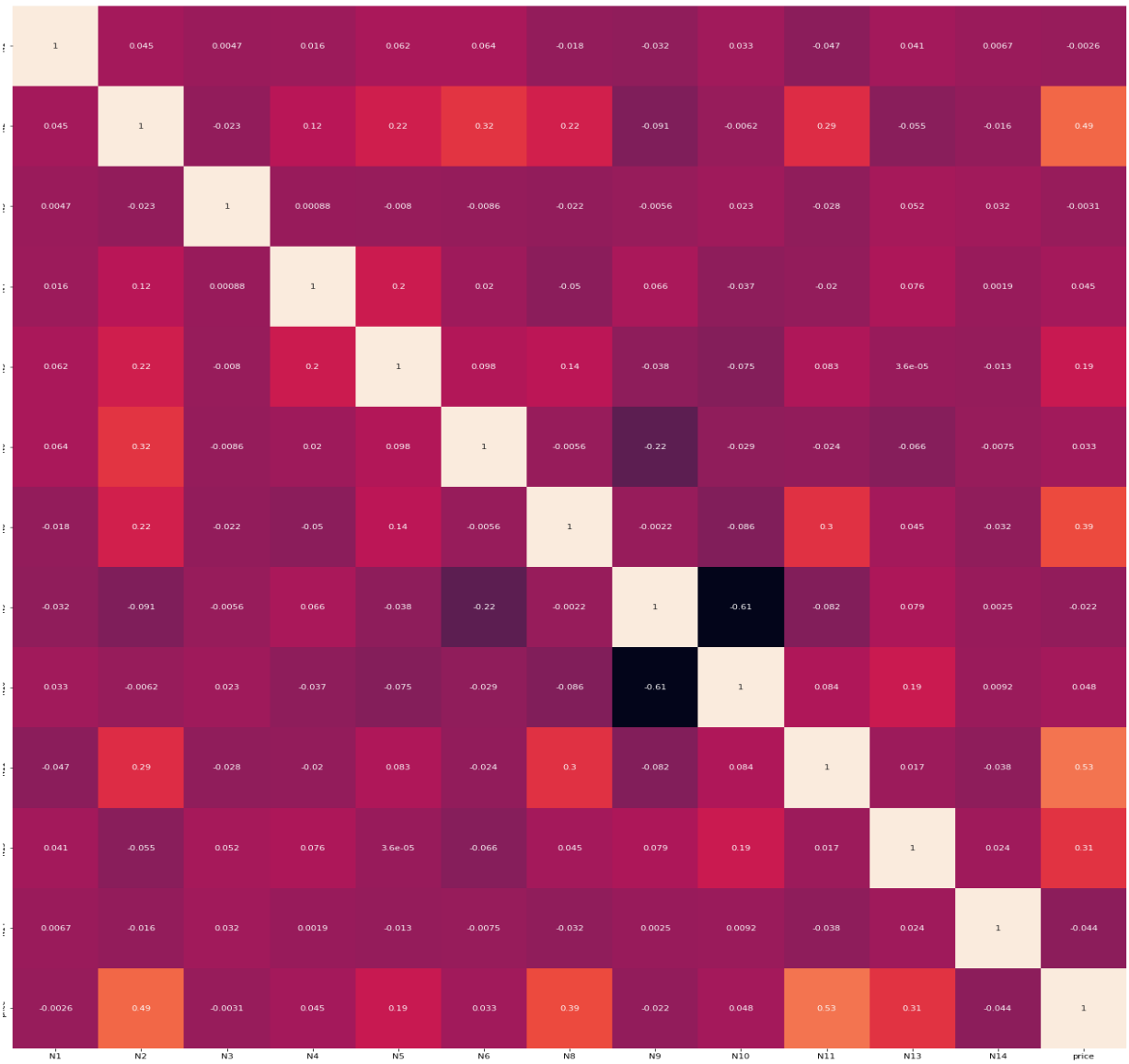


图 3.4

通过对 15 个匿名特征的相关性探索，发现匿名特征 N1 和 N3 *与价格*的相关性较小，所以我们对这几个匿名特征也进行了删除处理。

步骤四、填补缺失值。通过图 3.1，我们对缺失数据的特征进行填补，考虑到尽量降低训练集的噪声，我们对其采用了众数填补的方法，分别取每一个特征中数据的众数对空缺值进行填补。

3.2.2 数据转换

步骤一、对各个特征的数据分布进行绘图并分析，我们发现 price，newprice，N5 是明显的偏分布，为了使得下一步数据的归一化处理，我们对偏分布的特征数据进行了简单变换处理，取对数，使之近似的符合正态分布。

其中，Price，newprice，N5 特征的数据分布图分别对应下图 3.5，3.6，3.7。对 price 进行对数化处理后得到数据分布图 3.8.

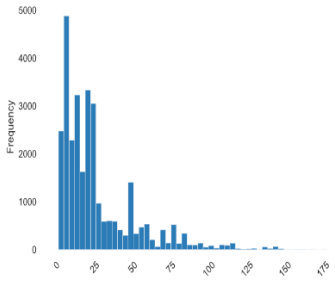


图 3.5

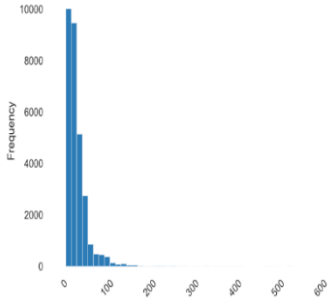


图 3.6

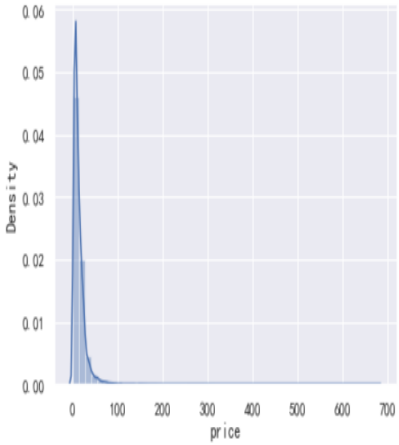


图 3.7

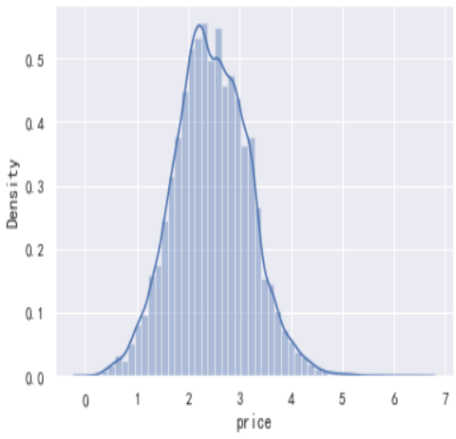


图 3.8

步骤二、对数据形式进行观察，我们发现除去特征 N11 外，其他的特征的数据类型都属于离散型和连续型，所以我们只对特征 N11 进行改造。通过我们的进一步观察，发现特征 N11 应符合离散型的特征，所以我们对它的数据进行属性构造，对应变化如下表。

原 N11 数据格式	改变后 N11 数据格式
‘1’	0
‘1+2’	1
‘3+2’	2

‘1+2，4+2’	3
‘1，3+2’	4
‘5’	5

四、问题一：建立二手车估价模型并预测

4.1 问题一的求解思路

问题一是对已有的二手车交易价格数据进行训练并对二手车交易价格进行预测的问题。在数据预处理中，我们能够得到二手车交易数据的一系列特点：首先是离散型和连续型数据同时存在，这就让我们无法通过简单的线性回归进行分析模拟。其次，交易数据中的特征比较多，维数比较大，数据量也是比较少，如果使用简单的决策树模型，很容易产生过拟合的现象，没有实际运用价值。综上考虑，我们采取了对 XGBoost 进行优化后的 lightGBM 算法。

相较于 XGboost 的方法，lightGBM 在决策树的生长策略上采用的是 leaf—wise 的生长策略，每次从当前所有叶子中找到分裂增益最大（一般也是数据量最大）的一个叶子，然后分裂，如此循环；但会生长出比较深的决策树，产生过拟合，所以，在 lightGBM 进行决策树的生长之前，我们采用了 CV 调参的方法，对之进行最大深度的限制，既保证了高效率又防止了过拟合。

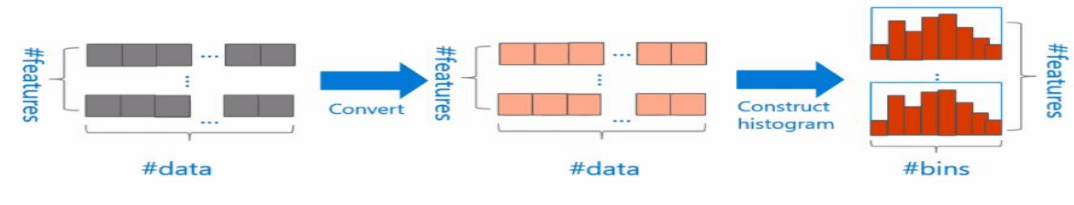
为了使我们的训练结果更加精确，我们尝试并使用了五折交叉验证的方法，通过尝试利用不同的训练集/测试集划分来对模型做多组不同的训练/测试，来应对测试结果过于片面以及训练数据不足的问题。

如此下来，我们选取的模型既防止了过拟合的产生又弥补了数据量较少的缺点，为我们的训练模型的建立提供了理论基础。

4.2 lightGBM 算法

LightGBM 算法在 GBDT 的基础上，先把连续的浮点特征值离散化成 k 个整数，同时构造一个宽度为 k 的直方图。在遍历数据的时候，根据离散化后的

值作为索引在直方图中累积统计量，当遍历一次数据后，直方图累积了数据的统计量，然后根据直方图的离散值，遍历寻找最优的分割点。示意图如下



这样可以就可以大大减少我们的训练时间，并且降低了我们的计算代价。

在直方图算法的基础上，lightGBM 还实现了 Goss 算法和 EFB 算法提高其效率。

4.2.1 Goss 算法

利用 Goss 算法，我们通过保存大梯度的二手车交易样本，随机选取小梯度的二手车交易样本，并为其弥补上一个常数的权重。如此一来，我们便可以弥补训练样本不足的缺点，同时也不会改变原有数据的信息。其具体的算法模式主要是先根据梯度对样本进行排序，选取占比为 a 的 top 样本，再从剩余数据中随机选取占比为 b 的样本，并乘以 g 的系数放大。通过证明，利用 Goss 算法，近似的误差很好，并且很贴近于使用所有数据的模型。

4.2.2 EFB 算法

在本题中，由于二手车的高维数据，同时又是比较稀疏的，可以采取一种损失最小的特征减少方法。并且，在稀疏特征空间中，许多特征都是互斥的，也就是它们几乎不能同时取到非 0 值。因此，我们可以安全的把这些互斥特征绑到一起形成一个特征，然后基于这些特征束构建直方图，这样可以加快我们的模型训练速度。

而 EFB 算法可以把很多特征绑到一起，形成更少的稠密特征束，这样可以避免对 0 特征值的无用的计算。这样的算法过程步骤一是找到允许特征之间的小冲突，这样的话，我们会得到更小的特征束数量，计算的效率会更高，让我们能够在效率和精度之间寻找平衡。

步骤二是合并寻找到的特征，降低训练的复杂程度，EFB 算法通过把互斥的特征放到不同的桶，从而构造一个特征束。

在进行完这两步之后，我们就对训练集的特征进行合理的划分，从而提高了精确度和效率。

4.3 五折交叉验证

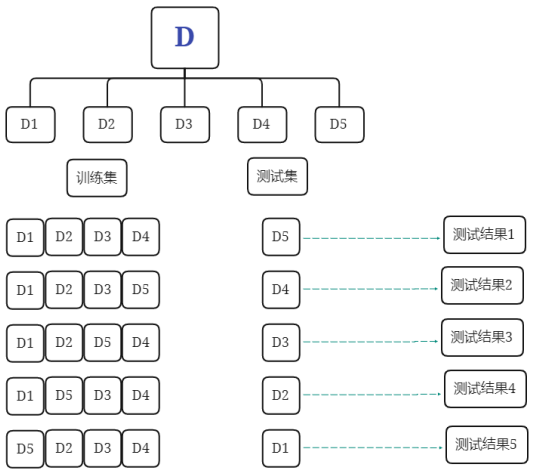
采用这种方法是利用不同训练集和测试集的划分对模型做出多组不同的训练和测试，解决训练结果片面和训练程度不足的问题。

交叉验证的做法为，将已有数据集分为比较均等且不相交的 k 份，采用数学表示：

$$D = D_1 \cup D_2 \cup \ldots \cup D_k$$

$$D_i \neq D_j (i \neq j)$$

然后我们取其中的一份进行测试，剩余的 k-1 份进行训练，然后求得最终 error 的平均值作为模型评价，帮助我们选取最优的方案。其流程图如下



在交叉验证的基础上，我们采取了比较普遍的五折交叉验证方法，先将我们的数据集分为 5 堆；然后我们选取一堆作为测试集，另外的 4 堆作为训练集；最后重复第二步五次，每次选取不同的训练集。经过五次交叉验证，得到最优的训练模型。

4.4 问题一的训练结果

我们将训练的过程分为两部分，第一部分是调参过程，我们首先将

lightGBM 中的参数做了最优处理，采用了 cv 调参的方法找到了最优的参数，其具体参数为

lightGBM 参数	
num_leaves	95
max_depth	7
max_bin	255
min_data_in_leaf	101
feature_fraction	1.0
bagging_fraction	1.0
bagging_freq	45
lambda_l1	1.0
lambda_l2	1.0
min_split_gain	1.0

第二部分将我们处理好的训练集进行训练，并对模型进行评估，其评估标准公式为：

相对误差 *Ape*：

$$Ape = |\hat{y} - y|/y \quad (1)$$

平均相对误差 *Mape*：

$$Mape = \frac{1}{m}\sum_{i=1}^{m} Ape_i \quad (2)$$

其中，真实值$y = (y_1, y_2, \ldots, y_m)$，预测模型值为 $\hat{y}$ =（ $\hat{y}_1$, $\hat{y}_2$, ⋯, $\hat{y}_m$ ）；

5%的误差准确率$Accuracy_5$ ：

$$Accuracy_5 = count(Ape \leq 0.05)/count(total) \quad (3)$$

其中，$count(Ape \leq 0.05)$为相对误差 Ape 在 5%以内的样本数量，$count$（$total$）为样本总数量。

最终得到精确度 $Accuracy$：

$$Accuracy = 0.2 * (1 - Mape) + 0.8 * Accuracy_5 \quad (4)$$

将我们处理后的训练数据集训练后，我们绘出了预测结果与原结果的二手车交易价格对比图，如下图 4.1。

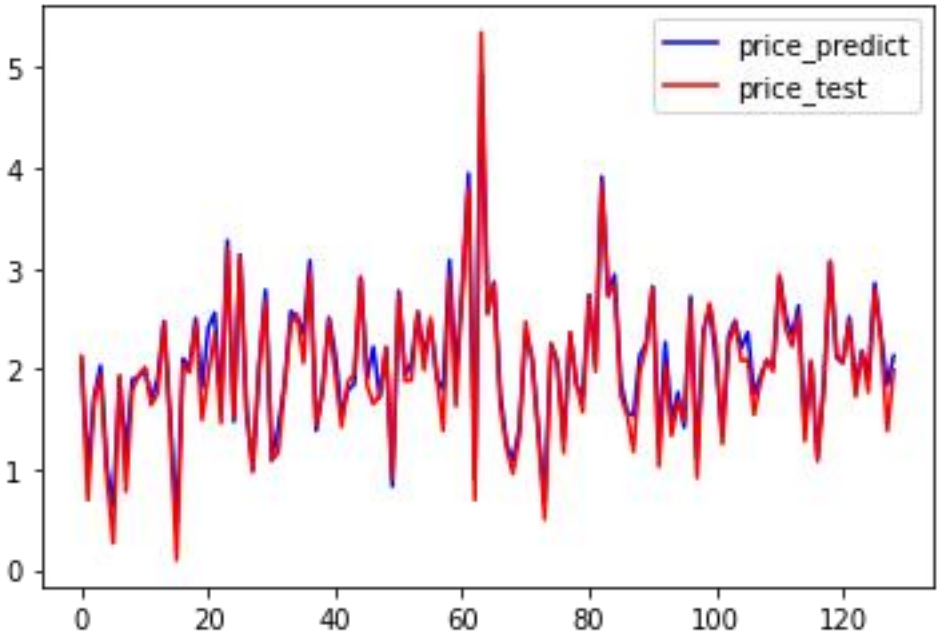


图 4.1

其中我们的模型评估结果为 0.83087.

五、问题二的数据预处理

5.1 原始数据集的介绍

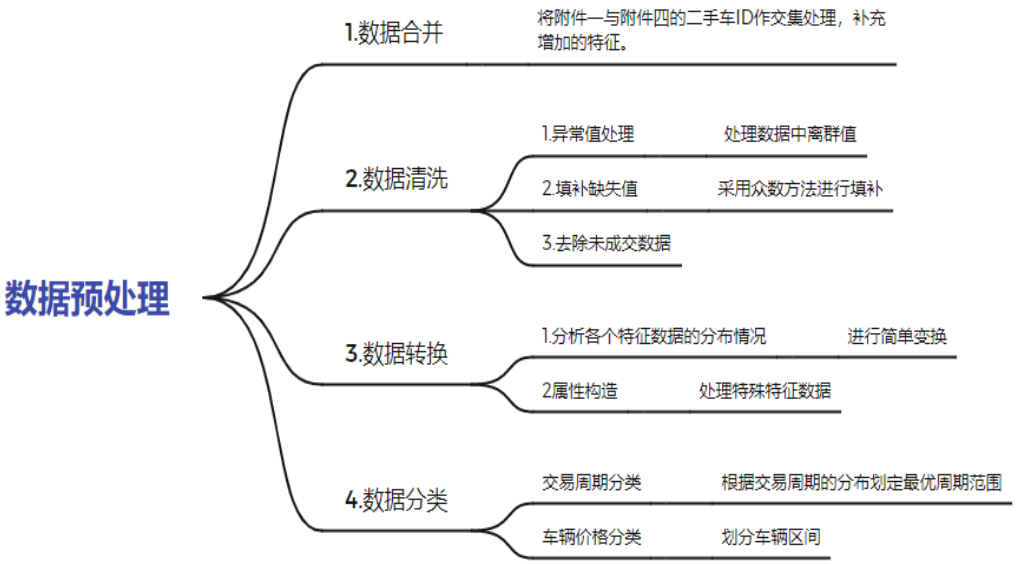
在给出的门店交易训练数据中，给出了对二手车交易周期的额外相关因素，其对应的特征和具体含义由下表给出

字段名	字段含义
Carid	二手车 ID

Pushdate	二手车的上架时间
Pushprice	二手车的上架价格
UpdatePriceTimeJson	二手车价格调整时间：调整后价格
PullDate	二手车下架时间
Withdrawdate	二手车成交时间
注：当车辆成交时，下架时间于成交时间相同	

在此问题的分析中，由于匿名特征的含义不明确，我们对附件一的匿名特征进行了剔除处理，仅仅分析已知特征。由于二手车成交时，下架时间和成交时间相同，所以我们对二手车的下架时间（pulldate）不进行考虑；同样的，题目要求我们作为定价师，所以原特征中的二手车上架价格对我们的分析没有意义，我们对这一特征同样不进行考虑。综上所述，在新增的特征中，我们只考虑二手车的上架时间（Pushdate），二手车的成交时间（withdrawdate），和最终成交价格 price。

对于问题二数据的处理过程，我们的思路如下



5.2 数据预处理

5.2.1 数据合并

我们先对门店训练数据（附件四）与估价训练数据（附件一）进行分析，发现门店训练数据种的车 ID 是估价训练数据车 ID 的子集，所以我们采取了取交集的方式，获得了对我们分析有效的数据。

5.2.2 数据清洗

对于异常值的处理。我们认为是当新车价和交易车价相差过于大时，我们认为这是不正常的交易情况，不具有普遍意义，应当作删除处理。对于新车价与成交价相差 100 以上的数据，我们都认为是异常值，也进行了删除处理。

对于缺失值的处理。与问题一的缺失值处理相同，我们同样采用取众数的方法进行填补，尽量减少数据中的噪声。

去除未成交的数据。因为我们需要分析影响成交周期的关键因素，所以我们就需要把未成交的数据进行剔除，得到对分析有效的数据。

5.2.3 数据转换

与问题一的数据分布分析相同，我们对数据的分布进行了分析，得到成交价格和新车价格的分布为偏分布，所以我们对之作简单变换，做对数处理，使之近似的符合正态分布，便于模型的训练。成交价格的数据分布如图 5.1，作对数处理后的数据分布如图 5.2。

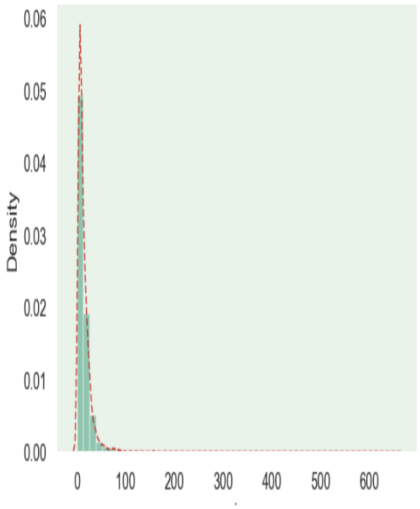


图 5.1

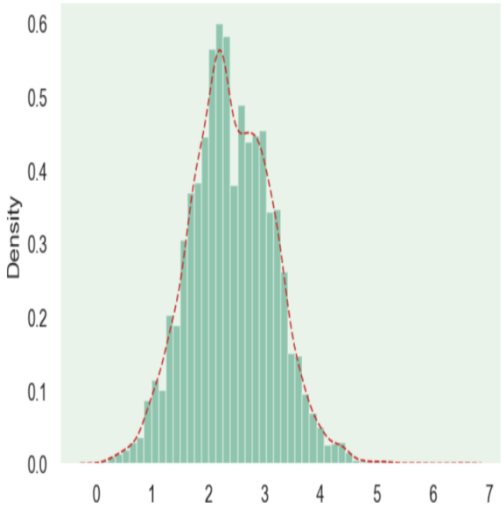


图 5.2

对于数据中的时间特征，同问题一的处理方式相同，提取出数据中的时间特征。

5.2.4 数据分类

对于交易周期的分类。首先我们对成交周期的数据分布进行分析，其数据分布如图 5.3.

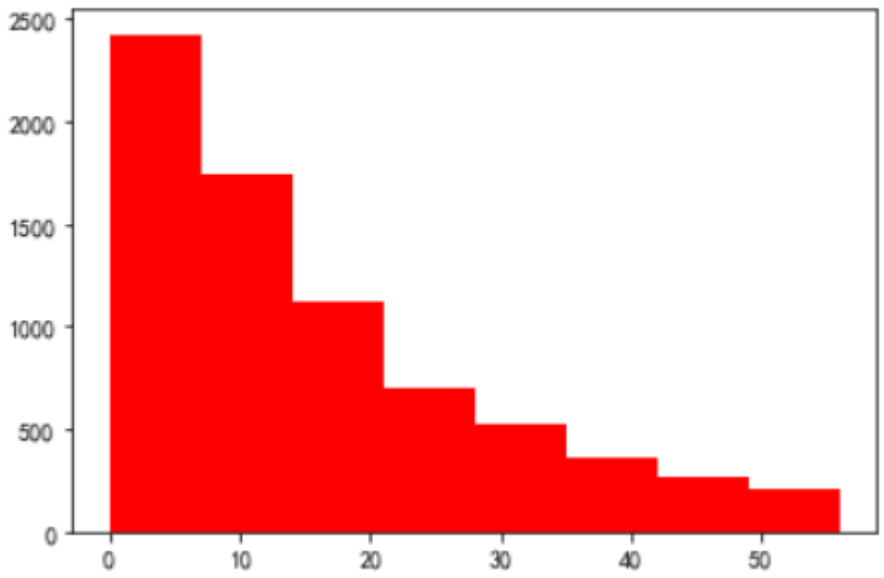


图 5.3

通过图中的数据，我们可以明显的观察出，成交周期主要集中在五十天以

内。考虑到训练集的数据较多可以达到比较好的训练结果，又根据相关的现实依据，我们认为，成交周期在两周以内是比较合理的交易周期，所以我们将成交周期在十二天以内划为一个理想区间，成交周期在十二天以外的划为一个改进区间。

对于车辆价格的分类。我们同样先对车辆价格的数据分布进行分析，其数据分布图如图 5.4.

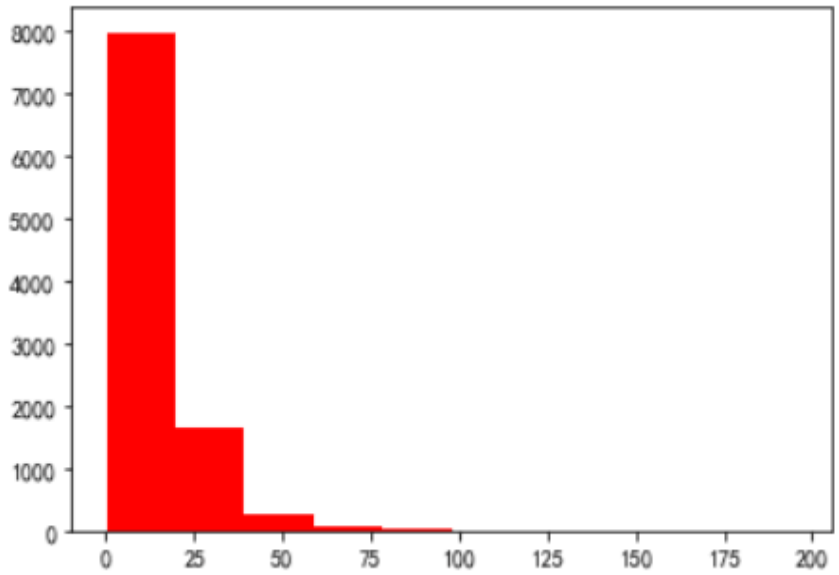


图 5.4

可以观察到，车辆的价格分布在 100（单位价格）以内，所以我们将车辆的价格分为三个区间。划分区间的依据是问题一中车辆的价格对应车辆的资产价值，所以我们以资产价值的高低进行划分。划分区间示意如下：

低资产价值车辆：

$$price < 25$$

中资产价值车辆：

$$25 \leq price \leq 50$$

高资产价值车辆：

$$50 < price$$

通过这三个区间的划分，我们可以得到不同资产价值车对应的不同模型，让我们对于数据结果的解释性更强。

六、问题二：挖掘二手车成交关键因素并给出建议

6.1 问题的求解思路

问题二要求我们解决两个问题：1.挖掘出二手车成交的关键因素。2.作为定价师改良价格以缩短成交周期。

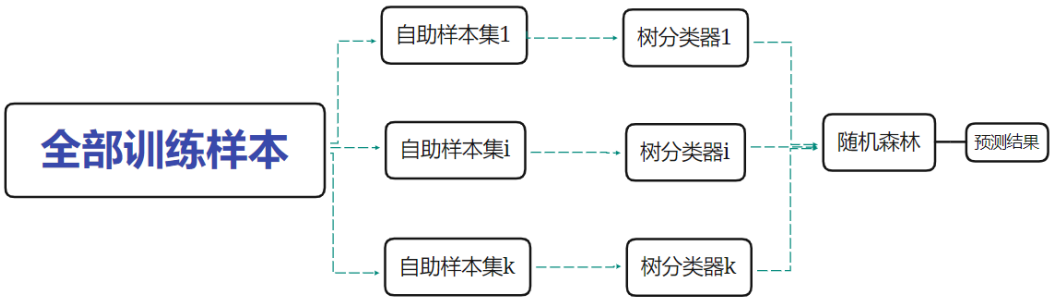
对于第一个问题，我们需要对所有成交的二手车数据进行权重分析，因为随机森林算法大致可以看成是从生成的多个决策树中挑选。所以在训练的过程中就会按照不同特征维度的先后划分方式来建立决策树。因此，最优的那颗树所对应的特征划分顺序也就代表着特征的重要程度。由于题目要求我们作为定价师进行分析，**我们认为，在给定的交易周期下，应该分析交易价格的影响因素，对交易价格影响大的因素可以证明也是影响交易周期大的因素。所以我们首先选出前 14 天内成交的数据作为训练集，把交易价格作为指标，对训练集进行训练，寻找权重因数，并对模型进行误差评估。**

对于第二个问题，我们首先选出了合理的交易周期，即 14 天之内成功交易。然后选取在 14 天内交易的数据进行训练，将训练的模型对超过 14 天才交易成功的数据进行价格预测，然后得到预测价格，这个价格在理论上可以让该二手车在 14 天内成功交易。依据这样的思路，我们既可以采用随机森林中的回归预测算法进行训练也可以采用传统的多元线性回归算法进行训练，考虑到选取最优的方案进行定价，我们分别采用了这两种算法进行训练。

6.2 随机森林算法

随机森林是一种基于集成学习的监督式机器学习算法。集成学习是一种学习类型，可以多次加入不同类型的算法或相同算法，以形成更强大的预测模型。随机森林结合了多个相同类型的算法，即多个决策的数目，所以称之为“随机森林”。由于其不必担心过度拟合，分类速度很快，能够高效处理大样本的数据，抗噪音能力比较强等特点。在对分析影响二手车成交周期的关键因素中，能够估计哪个特征在分类中更重要，进而得到关键因素。

为了进一步说明随机森林的原理，绘制随机森林的流程图如下



其过程为:

步骤一，从数据集中随机选择 N 个样本子集；

步骤二,基于这 N 个样本子集构建决策树。；

步骤三选择算法中所需树的棵数，然后重复步骤 1 和 2。；

通过自主抽样的技术，由回归树构成组合模型，该模型将数值型变量作为预测变量，可以生成多元的非线性回归随机森林模型。

模型评估参数：

设预测值：$\hat{y}=(\hat{y}_1,\hat{y}_2,\cdots,\hat{y}_m)$；真实值：$y=(y_1,y_2,\ldots,y_m)$;

均方误差（MSE）：

$$\mathrm{MSE}=\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i-y_i)^2$$

平均绝对误差（MAE）：

$$\mathrm{MAE}=\frac{1}{m}\sum_{i=1}^{m}|\hat{y}_i-y_i|$$

均方根误差（RMSE）：

$$RMSE=\sqrt{MSE}$$

6.3 多元回归算法

多元回归模型是解决多特征的数据的一种比较基础的做法，他的优点在于

解释性比较强，在解决线性问题上表现比较好。

多元线型回归模型的一般形式为

$$Y_i = \beta_0 + \sum_{i=1}^{n} \beta_i X_{ki} + u_i$$

其中，β_0称为回归常数，β_i称为回归系数，X_{ki}为解释变量，Y为被解释变量，k为解释变量的数目。

在这个问题中，被解释变量 Y为成交价格，解释变量为我们已知的特征，如新车价、国别等

6.4 影响二手车交易周期的关键因素

我们将在 14 天以内成交的低资产价值二手车的交易数据利用随机森林进行训练，分析特征对价格影响的权重，并进行排序，得到下表：

特征字段	权重
Newprice	0.338917
Displacement	0.132434
LicenseDate_year	0.089950
Modelyear	0.072864
Registerdate_year	0.072319
Country	0.048233
Gearbox	0.045388
Maketype	0.035014
Brand	0.025909
Mileage	0.023462
Serial	0.020305
Carcode	0.018256
Model	0.009749
Color	0.007308
LicenseDate_day	0.007009

cityid	0.006444
Tradetime_day	0.006443
Tradetime_month	0.005486
Registerdate_month	0.005282
licenseDate_month	0.005206
seatings	0.004849
oiltype	0.002891
transfercount	0.002121
Tradetime_year	0.001552
Registerdate_day	0.000000(数值极小)
Trans_category	0.000000(数值极小)
Car_catergory	0.000000(数值极小)

对我们的模型进行评估得下表：

模型评估参数	数值
MAE	0.13762973051190885
MSE	0.05202231303077569
RMSE	0.22808400432905349

根据模型得评估结果，我们利用随机森林进行训练比较理想，误差非常的小，对我们得到的结果具有较强的说服力。(用同样的做法分析中资产价值车和高资产价值车，我们得到的结果相差不大，在这里我们只给出了低资产价值车的分析数据，其他数据在支撑材料中)。通过分析权重，我们得到影响交易周期的关键因素为，新车价格。

6.5 给出定价方案和预期效果

6.5.1 随机森林回归

在 6.4 得到的训练模型基础上，我们对 14 天以后成交的车辆进行预测，使调整后的价格能满足在 14 天以内成交的预期效果（调整后定价价格数据在支撑材料中）。然后我们画出原价格和修改后价格的对比图。

其中低资产价值车的对比图为图 6.1，中资产价值车的对比图为图 6.2，高资产价值车的对比图为图 6.3。

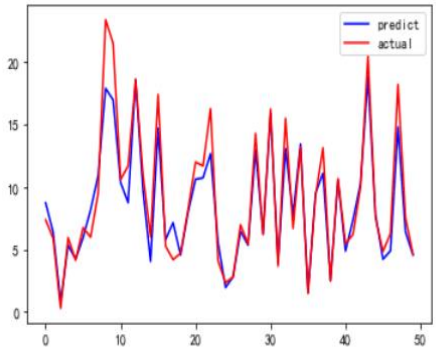


图 6.1

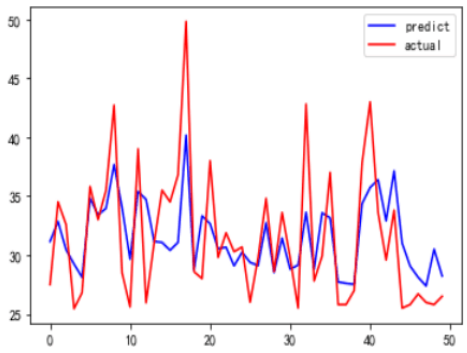


图 6.2

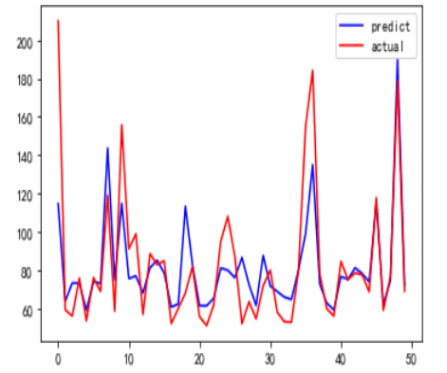


图 6.3

通过我们的对比图可以看到，在低资产价值车和中资产价值车的预测定价价格比较合理，而在于高资产价值车中，发现误差比较的大，经过我们的分析可得，有极大的可能是因为高资产价值车的训练数据比较少所导致的。

6.5.2 多元线性回归

与随机森林回归的思路相同，我们在 5.2 数据预处理得到的数据中，对 14 天以内的成交的二手车交易数据进行回归，得到多元线性回归方程，然后将方程用于交易周期超出 14 天的数据进行预测（调整后定价价格数据在支撑材料中），然后绘制出修改后价格和原价格的对比图。

其中低资产价值车的对比图为图 6.4，中资产价值车的对比图为图 6.5，高资产价值车的对比图为图 6.6。

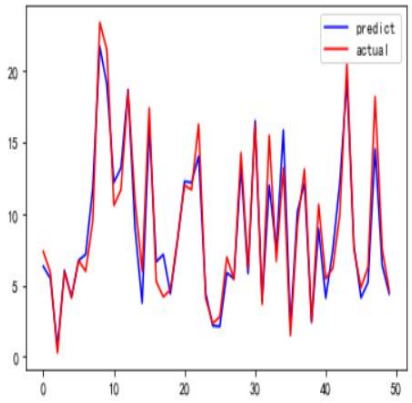


图 6.4

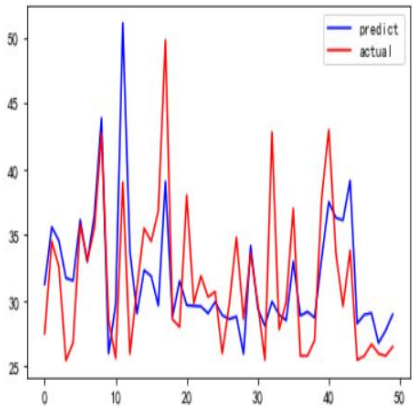


图 6.5

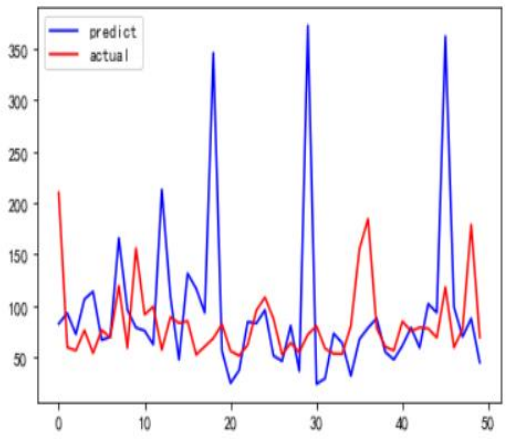


图 6.6

可以看出，利用多元线性回归模型，在低资产价值车预测定价价格比较合理，相比于随机森林，具有很强的解释性优势。而对于中高资产价值车而言，使用多元线性回归的方法数据并不是很合理，这可能是由于交易周期与定价价格并不是简单的线性相关的缘故。同时，在低资产价值车中，我们也可以发现

随机森林的预测定价价格是要比多元线性回归的结果要好，更为合理。

6.5.3 总结方案

根据随机森林回归和多元线性回归得到的结果，进行比较分析，我们在低中资产价值车的定价方案中，采取随机森林回归的模型结果进行预测，而对于高资产价值车的定价方案中，两种回归模型效果都不太好，但由我们分析认为，在高资产价值车的定价方案中，随机森林训练结果不好的原因是由于训练数据比较少的缘故，通过增加训练数据，可以改善，所以在高资产价值车的定价方案中，我们依然采用随机森林回归的模型结果进行预测。

在这种方案下，我们理论上可以将车的交易周期全部置入 14 天以内，但这是有条件的，必须假设所有展出的二手车都能够被购买，而对于往往不受顾客青睐的二手车车型，购买成交率很低，运用此方案得到的定价并不能实现在 14 天内完成交易，达不到预期效果。

七、问题三：分析二手车成交与否的关键因素。

7.1 问题来源

在解决问题二的过程中，我们发现作为定价师给出定价方案的过程中，我们的必须要假设所有的展销二手车能够被客户购买，这就让我们思考影响二手车能不能被客户青睐的关键因素是什么，这对企业选择销售策略具有重要的意义。对于高成交率的车型，可以避免二手车最后卖不出去选择批量低价出售所带来的经济损失，并且能够在一定程度上分析出当今二手车市场上深受顾客青睐的车型是什么，根据分析结果，在 020 的商业模式下，可以最大化满足客户需求。

7.2 问题解决思路

首先，我们需要定义数据中的成交与未成交的特征，定义如下：

构造特征：成交与否	
数据形式	含义
0	未成交
1	成交

对于是否成交的判断，我们根据门店交易训练集中成交时间的有无进行判断。具体判断如下：

有成交时间：成交

无成交时间：未成交

然后，我们利用在 5.2 中的处理后得到的数据集，将与门店交易训练集中的特征相同的进行删除，对应合并我们新定义的特征：成交与否，得到解决问题三的数据集。

在解决问题二的过程中，我们使用了随机森林的回归模型，这启发我们同样利用随机森林解决问题三。由于随机森林不仅可以进行回归预测还可以进行类别预测，所以我们将处理好的数据集代入随机森林的类别模型中进行训练，和第二问的思路相同，求出权重系数并给出模型评估。

7.3 问题三解决结果

将新的训练集代入模型进行训练后，我们得到的权重系数如下表：

特征字段	权重
Mileage	0.078992
Price	0.077869
Model	0.075827
Newprice	0.066304
Tradetime_day	0.064982
serial	0.063006

licenseDate_day	0.062473
cityId	0.052439
tradeTime_month	0.046759
brand	0.045609
licenseDate_month	0.043546
gearbox	0.043519
modelyear	0.035886
displacement	0.032944
Registerdate_year	0.031915
color	0.031705
licenseDate_year	0.030747
country	0.024274
tranferCount	0.017064
Tradetime_year	0.013458
carcode	0.010290
maketype	0.009473
seatings	0.007347
oiltype	0.002317
Registerdate_day	0.000000(数值极小)
Car_category	0.000000(数值极小)

其中，我们的模型评估结果为：

模型评估参数	评估结果
ROC	0.90
Logloss	1.51

经过分析，我们得到影响二手成交与否的关键因素主要为，行驶里程，估价价格和车辆车型。

参考文献

[1] zjuPeco,利用随机森林对特征重要性进行评估,https://blog.csdn.net/zjuPeco/article/details/77372645，2022 年 1 月 17 日.

[2] Maple,二分类评估指标 F1&AUC&LogLoss，https://zhuanlan.zhihu.com/p/87260891，2022 年 1 月 17 日.

[3] 5 号程序员,机器学习：Python 常用库——Statsmodels 库，https://zhuanlan.zhihu.com/p/260701846,2022 年 1 月 15 日.

[4] 肥波喇齐,使用 pandas-profiling 生成数据的详细报告，https://zhuanlan.zhihu.com/p/47548106,2022 年 1 月 14 日.

[5] CC_且听风吟,初探监督学习 2：使用线性回归预测波士顿房价（回归模型），https://blog.csdn.net/weixin_43826242/article/details/88996092,2022 年 1 月 17 日.

[6] 浅笑古今,LightGBM 调参笔记，https://blog.csdn.net/u012735708/article/details/83749703，2022 年 1 月 16 日.

[7] 张月.北京二手车保值率的影响因素分析[D].天津财经大学,2019.DOI:10.27354/d.cnki.gtcjy.2019.000146.

[8] 李原吉,李丹,唐淇.基于 XGBoost 算法的依据车辆信息预估二手车成交价格模型[J].电子测试,2021(21):47-49.DOI:10.16520/j.cnki.1000-8519.2021.21.016.

[9]王静娜.基于随机森林算法的二手车估价模型研究[D].北京交通大学,2019.DOI:10.26944/d.cnki.gbfju.2019.000864.

[10]华金虎,冯梓豪.基于贝叶斯优化算法的 LightGBM 房产价值评估模型[J].中国科技信息,2022(01):112-114.

附录

使用软件名称：Pycharm

问题一的源程序：

```
#导入用到的库
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import StratifiedKFold, KFold
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
import lightgbm as lgb
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
import pandas as pd
import warnings
from sklearn.preprocessing import scale
from sklearn.model_selection import cross_val_score
import lightgbm as lgb
from sklearn.model_selection import KFold
import xgboost as xgb
from catboost import CatBoostRegressor
import time
import numpy as np
from sklearn.preprocessing import StandardScaler
# 2、获取并载入数据
train = pd.read_csv('train_clear (1).csv')
test = pd.read_csv('test_clear (1).csv')
```

```
n_price = 75
train_X = train[train['price'] < n_price]
# 使用对数的右偏变换函数，将数据分布转为近似正态分布
train_y = train_X['price']
del train_X['price']
scaler = StandardScaler()  # 归一化处理
train_x = scaler.fit_transform(train_X)
test_x = scaler.fit_transform(test)

# 训练用的字典
params = {'learning_rate': 0.01,
                'boosting_type': 'gbdt',
                'objective': 'regression_l1',
                'metric': 'mae',
                'min_child_samples': 46,
                'min_child_weight': 0.01,
                'feature_fraction': 1.0,
                'bagging_fraction': 1.0,
                'bagging_freq': 45,
                'num_leaves': 95,
                'max_depth': 7,
                'n_jobs': -1,
                'seed': 2019,
                'verbose': -1,
                'max_bin': 255,
                'min_data_in_leaf':101,
                'lambda_l1': 1.0, 'lambda_l2': 1.0,
                'min_split_gain': 1.0}

# 定义准确性函数
def Accuracy_metric(y_true, y_pred):
```

```
    n = len(y_true)

    mape = sum(np.abs((y_true - y_pred)/y_true))/n

    Apexiaoyu005 = pd.DataFrame(abs(y_true - y_pred)/y_true)

    Accuracy = (Apexiaoyu005[Apexiaoyu005 <= 0.05].count() /

    Apexiaoyu005.count())*0.8+0.2*(1-mape)

    return float(Accuracy)

# 训练模型
val_pred = np.zeros(len(train_x))
val_true = np.zeros(len(train_x))
preds = np.zeros(len(test_x))
folds = 5
# seeds = [1234]
# for seed in seeds:
kfold = KFold(n_splits=folds, shuffle=True, random_state=4321)
for fold, (trn_idx, val_idx) in enumerate(kfold.split(train_x, train_y)):
    print('fold ', fold + 1)
    x_trn, y_trn, x_val, y_val = train_x[trn_idx], train_y.iloc[trn_idx], train_x[va
l_idx], train_y.iloc[val_idx]
    train_set = lgb.Dataset(x_trn, y_trn)
    val_set = lgb.Dataset(x_val, y_val)
    model = lgb.train(params, train_set, num_boost_round=5000, valid_sets=(trai
n_set, val_set), early_stopping_rounds=500,
                      verbose_eval=False)
    val_pred[val_idx] += model.predict(x_val, predict_disable_shape_check=True)
    preds += model.predict(test_x, predict_disable_shape_check=True) / folds
    val_true[val_idx] += y_val
```

```
Accuracy = Accuracy_metric(val_true, val_pred)
c = float(Accuracy)
print('-'*120)
print('Accuracy ', round(c, 5))
pd.DataFrame(preds).to_csv('估价模型结果.csv', header=0)

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

LightGBM CV 调参源程序：

```
import pandas as pd
import lightgbm as lgb
from sklearn import metrics
import numpy as np
from sklearn.model_selection import train_test_split

df=pd.read_csv('train_clear (2).csv')


y=df['price']
df=df.drop(columns='price')
X=df
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size
=0.2)

### 数据转换
print('数据转换')
lgb_train = lgb.Dataset(X_train, y_train, free_raw_data=False)
lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train, free_raw_data=False)

### 设置初始参数--不含交叉验证参数
print('设置参数')
```

```
params = {
    'boosting_type': 'gbdt',
    'objective': 'regression_l1',
    'metric': 'auc',
    'nthread': 4,
    'learning_rate': 0.1
}

### 交叉验证(调参)
print('交叉验证')
max_auc = float('0')
best_params = {}

# 准确率
print("调参 1：提高准确率")
for num_leaves in range(5, 100, 5):
    for max_depth in range(3, 8, 1):
        params['num_leaves'] = num_leaves
        params['max_depth'] = max_depth

        cv_results = lgb.cv(
            params,
            lgb_train,
            seed=1,
            nfold=5,
            metrics=['auc'],
            early_stopping_rounds=10,
            verbose_eval=True,
            stratified=False

        )
```

```
            mean_auc = pd.Series(cv_results['auc-mean']).max()
            boost_rounds = pd.Series(cv_results['auc-mean']).idxmax()

            if mean_auc >= max_auc:
                max_auc = mean_auc
                best_params['num_leaves'] = num_leaves
                best_params['max_depth'] = max_depth
if 'num_leaves' and 'max_depth' in best_params.keys():
    params['num_leaves'] = best_params['num_leaves']
    params['max_depth'] = best_params['max_depth']

# 过拟合
print("调参 2：降低过拟合")
for max_bin in range(5, 256, 10):
    for min_data_in_leaf in range(1, 102, 10):
        params['max_bin'] = max_bin
        params['min_data_in_leaf'] = min_data_in_leaf

        cv_results = lgb.cv(
            params,
            lgb_train,
            seed=1,
            nfold=5,
            metrics=['auc'],
            early_stopping_rounds=10,
            verbose_eval=True,
            stratified=False
        )

        mean_auc = pd.Series(cv_results['auc-mean']).max()
        boost_rounds = pd.Series(cv_results['auc-mean']).idxmax()
```

```
            if mean_auc >= max_auc:
                max_auc = mean_auc
                best_params['max_bin'] = max_bin
                best_params['min_data_in_leaf'] = min_data_in_leaf
if 'max_bin' and 'min_data_in_leaf' in best_params.keys():
    params['min_data_in_leaf'] = best_params['min_data_in_leaf']
    params['max_bin'] = best_params['max_bin']

print("调参 3：降低过拟合")
for feature_fraction in [0.6, 0.7, 0.8, 0.9, 1.0]:
    for bagging_fraction in [0.6, 0.7, 0.8, 0.9, 1.0]:
        for bagging_freq in range(0, 50, 5):
            params['feature_fraction'] = feature_fraction
            params['bagging_fraction'] = bagging_fraction
            params['bagging_freq'] = bagging_freq

            cv_results = lgb.cv(
                params,
                lgb_train,
                seed=1,
                nfold=5,
                metrics=['auc'],
                early_stopping_rounds=10,
                verbose_eval=True,
                stratified=False
            )

            mean_auc = pd.Series(cv_results['auc-mean']).max()
            boost_rounds = pd.Series(cv_results['auc-mean']).idxmax()

            if mean_auc >= max_auc:
                max_auc = mean_auc
```

```
                    best_params['feature_fraction'] = feature_fraction
                    best_params['bagging_fraction'] = bagging_fraction
                    best_params['bagging_freq'] = bagging_freq

if 'feature_fraction' and 'bagging_fraction' and 'bagging_freq' in best_params.keys
():
    params['feature_fraction'] = best_params['feature_fraction']
    params['bagging_fraction'] = best_params['bagging_fraction']
    params['bagging_freq'] = best_params['bagging_freq']

print("调参 4：降低过拟合")
for lambda_l1 in [1e-5, 1e-3, 1e-1, 0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0]:
    for lambda_l2 in [1e-5, 1e-3, 1e-1, 0.0, 0.1, 0.4, 0.6, 0.7, 0.9, 1.0]:
        params['lambda_l1'] = lambda_l1
        params['lambda_l2'] = lambda_l2
        cv_results = lgb.cv(
            params,
            lgb_train,
            seed=1,
            nfold=5,
            metrics=['auc'],
            early_stopping_rounds=10,
            verbose_eval=True,
            stratified=False
        )

        mean_auc = pd.Series(cv_results['auc-mean']).max()
        boost_rounds = pd.Series(cv_results['auc-mean']).idxmax()

        if mean_auc >= max_auc:
            max_auc = mean_auc
            best_params['lambda_l1'] = lambda_l1
```

```
                best_params['lambda_l2'] = lambda_l2
if 'lambda_l1' and 'lambda_l2' in best_params.keys():
    params['lambda_l1'] = best_params['lambda_l1']
    params['lambda_l2'] = best_params['lambda_l2']

print("调参 5：降低过拟合 2")
for min_split_gain in [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]:
    params['min_split_gain'] = min_split_gain

    cv_results = lgb.cv(
        params,
        lgb_train,
        seed=1,
        nfold=5,
        metrics=['auc'],
        early_stopping_rounds=10,
        verbose_eval=True,
        stratified=False
    )

    mean_auc = pd.Series(cv_results['auc-mean']).max()
    boost_rounds = pd.Series(cv_results['auc-mean']).idxmax()

    if mean_auc >= max_auc:
        max_auc = mean_auc

        best_params['min_split_gain'] = min_split_gain
if 'min_split_gain' in best_params.keys():
    params['min_split_gain'] = best_params['min_split_gain']

print(best_params)
```

##

使用软件：jupyter

问题二源代码：

回归模型：

导入需要的库

```
import scipy.stats as st
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from pylab import mpl
import numpy as np
mpl.rcParams['font.sans-serif'] = ['SimHei'] # 指定默认字体
mpl.rcParams['axes.unicode_minus'] = False
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from tqdm import tqdm
tqdm.pandas()
# sns.set(style='darkgrid')
```

2、读取数据

```
df1 = pd.read_csv('file1.csv')
df4 = pd.read_csv('file4.csv')
```

数据处理

```
df_11 = df1[['carID','price']]
df4 = df_11.merge(df4,on='carID')
df_trans = df4[df4.withdrawDate.notna()]
# 计算交易周期
df_trans['pushDate'] = pd.to_datetime(df_trans['pushDate'])
```

```
df_trans['withdrawDate'] = pd.to_datetime(df_trans['withdrawDate'])
trans_circle = pd.DataFrame(df_trans['withdrawDate'] -
df_trans['pushDate'])
df_trans['transcycle'] = trans_circle[0]
# 转为整型
sep = df_trans['transcycle'].astype(str).str.split(' ', expand=True)
df_trans['transcycle'] = sep[0].astype(int)
df4['transcycle'] = sep[0].astype(int)
# 取 file4 中与 file1 中相同 carid 的数据
trans_circle_info = df1[df1.carID.isin(df4.carID.tolist())]
# trans_circle_info.info()
df5 = df4[['carID','transcycle']]
NEW_trans_circle = trans_circle_info.merge(df5,on='carID')
#  交易周期分桶
# 分成三类，没卖出去，为一类
NEW_trans_circle.loc[NEW_trans_circle[(NEW_trans_circle['transcycle']
<=14)].index, 'trans_category'] = 1
NEW_trans_circle.loc[NEW_trans_circle[(NEW_trans_circle['transcycle']>
14)].index, 'trans_category'] = 2
# 没卖出去的，交易周期编码为 0
NEW_trans_circle['trans_category'] =
NEW_trans_circle['trans_category'].fillna(0)
# 缺失值处理，以众数填充
NEW_trans_circle['carCode'] = NEW_trans_circle['carCode'].fillna(1)
NEW_trans_circle['maketype'] = NEW_trans_circle['maketype'].fillna(2)
plt.title('Normal')
sns.distplot(np.log1p(NEW_trans_circle['mileage']), kde=False,
fit=st.norm)
plt.savefig('1.png')
cols = ["carID", "tradeTime", "brand", "serial", "model", "mileage",
"color", "cityId", "carCode", "transferCount", "seatings",
"registerDate",
```

```
        "licenseDate",     "country",     "maketype",     "modelyear",
"displacement",        "gearbox",        "oiltype",        "newprice",
"price","trans_category"]
df = NEW_trans_circle[cols]
#  缺失值处理
# 以下分类特征全部填充众数
df['carCode'] = df['carCode'].fillna(1)
df['modelyear'] = df['modelyear'].fillna(2017)
df['country'] = df['country'].fillna(779412)
df['maketype'] = df['maketype'].fillna(2)
df['gearbox'] = df['gearbox'].fillna(3)
# # 时间处理(提取年月日)
df['tradeTime'] = pd.to_datetime(df['tradeTime'])
df['registerDate'] = pd.to_datetime(df['registerDate'])
df['licenseDate'] = pd.to_datetime(df['licenseDate'])



df['tradeTime_year'] = df['tradeTime'].dt.year
df['tradeTime_month'] = df['tradeTime'].dt.month
df['tradeTime_day'] = df['tradeTime'].dt.day

df['registerDate_year'] = df['registerDate'].dt.year
df['registerDate_month'] = df['registerDate'].dt.month
df['registerDate_day'] = df['registerDate'].dt.day

df['licenseDate_year'] = df['licenseDate'].dt.year
df['licenseDate_month'] = df['licenseDate'].dt.month
df['licenseDate_day'] = df['licenseDate'].dt.day

del df['tradeTime']
del df['registerDate']
del df['licenseDate']
```

```
y_p = df[df['price'] <= 200]
## 3) 查看预测值的具体频数
plt.hist(y_p['price'], orientation='vertical',
         histtype='bar', color='red')
plt.savefig('price_bar1.png')
plt.show()
# # 车辆分类
df.loc[df[df['price']<= 25].index, 'car_category'] = 0 # 低端车
df.loc[df[(df['price']> 25)&(df['price']<= 50)].index, 'car_category']
= 1 # 中端车
df.loc[df[df['price']> 50].index, 'car_category'] = 2 # 高端车
# # 转换数据分布
df_a = df.copy()
df_a['price'] = np.log1p(df_a['price'])
df_a['newprice'] = np.log1p(df_a['newprice'])
cols = list(df)
cols.insert(0,cols.pop(cols.index('price'))) # 2 是将 d 放在哪一列,
cols.pop(cols.index('d')) 是要换的 d 列
df_a = df_a.loc[:,cols] # 开始按照两列互换
# # 存储将要回归分析的数据
df_a[df_a['trans_category']==0].to_csv('没卖出.csv', index=0)
df_a[df_a['trans_category'] == 1].to_csv('前 14 天内卖出.csv', index=0)
df_a[df_a['trans_category'] == 2].to_csv('14 天之后卖出.csv', index=0)

# # 对分类的样本可视化
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA

# 只举例一个数据
df1 = pd.read_csv('前 14 天内卖出.csv')
data1 = np.array(df1)
```

```
pred0 = list(df1['car_category'])


pca = PCA(n_components=2)  # 输出两维
newData1 = pca.fit_transform(data1)  # 载入 N 维

x1, y1 = [], []
x2, y2 = [], []
x3, y3 = [], []
for index, value in enumerate(pred0):
    if value == 0:
        x1.append(newData1[index][0])
        y1.append(newData1[index][1])
    elif value == 1:
        x2.append(newData1[index][0])
        y2.append(newData1[index][1])
    elif value == 2:
        x3.append(newData1[index][0])
        y3.append(newData1[index][1])
plt.figure(figsize=(10, 10))
plt.scatter(x1, y1)
plt.scatter(x2, y2)
plt.scatter(x3, y3)
plt.xticks(fontsize=12, fontweight='bold')  # 默认字体大小为 10
plt.yticks(fontsize=12, fontweight='bold')
plt.savefig('前 14 天卖出的数据分布.png')
plt.show()
```

多元回归（只列举低端车）

```
import pandas as pd
import numpy as np

week1_data_1 = pd.read_csv('前 14 天内卖出.csv')
```

```
# 建立回归低端车的方程
category1_data = week1_data_1[week1_data_1['car_category']==0]

X1 = category1_data
y1  = category1_data['price']
del X1['price']

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
X_train,X_test,y_train,y_test=train_test_split(X1,y1,test_size=.9,ran
dom_state=0)
linreg=LinearRegression()
model=linreg.fit(X_train,y_train)
y_pred=linreg.predict(X_test)

plt.figure()
plt.plot(range(len(y_pred[1:130])),y_pred[1:130],'b',label="price_pre
dict")
plt.plot(range(len(y_pred[1:130])),y_test[1:130],'r',label="price_tes
t")
plt.legend(loc="upper right")
plt.savefig('（低端）predict.png')
plt.show()

from statsmodels import api as sm
# 输出多元回归方程的系数

X= sm.add_constant(X1)
result = sm.OLS(y1,X1).fit()
result.params
# 用前 14 天的方程去回归后 14 天的 price
```

```
model1 = linreg.fit(X1,y1)
# 第一个类别的回归价格
week2_data_1 = pd.read_csv('14 天之后卖出.csv')
category1_data_week2 = week2_data_1[week2_data_1['car_category']==0]
X2 = category1_data_week2
y2  = category1_data_week2['price']
del X2['price']
cluter1_price = model1.predict(X2)
x = range(50)
plt.plot(x,np.expm1(cluter1_price[0:50]),color='b',label='predict')
plt.plot(x,np.expm1(y2[0:50]),color='r',label='actual')
plt.legend()
plt.savefig('（低端）14 天后 predict.png')
# 把预测的价格与 carID 对应保存，生成文件“低端车调整后的价格.csv”
df3 = category1_data_week2.carID
a1 = np.array(df3)
a2 = list(zip(a1,cluter1_price))
result = pd.DataFrame(a2)
result.columns=['carID', 'price']
# 把价格转换回取对数之前的数据
result['price']=np.expm1(result['price'])
result.to_csv('低端车调整后的价格.csv')
```

随机森林模型（数据预处理与回归模型相同，下面只给模型代码）：

```
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier
df1 = pd.read_csv('./data/sale/第二周之内卖出.csv')
#转换类型，以低端车为例
data2 = df1[df1['car_category']==0]
data1=pd.DataFrame(data2)
#划分训练集和测试集
```

```
x, y = data1.iloc[:, 1:].values, data1.iloc[:, 0].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.2, random_state = 0)
feat_labels = data1.columns[1:]
#标准化
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
from sklearn.ensemble import RandomForestRegressor
regressor =
RandomForestRegressor(n_estimators=200,random_state=42,max_features=6
,n_jobs=-1)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
#模型评估
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,
y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,
y_pred))
print('Root Mean Squared Error:',
      np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
importances = regressor.feature_importances_
indices = np.argsort(importances)[::-1]
for f in range(x_train.shape[1]):
print("%2d) %-*s %f" % (f + 1, 30, feat_labels[indices[f]],
importances[indices[f]]))
#举例 训练好模型后
#求相对于数据 x 的预测值 y_pred
```

```
x= sc.transform(x)
y_pred = regressor.predict(x)
y_1=pd.DataFrame(y_pred)
y_1['carid']=list(data2['carid'])
y_1['carid'].isnull().sum()
cols = list(y_1)
cols.insert(0,cols.pop(cols.index('carid')))
y_1 = y_1.loc[:,cols] # 开始按照两列互换
y_1.info()
y_1=y_1.rename(columns = {0:'price'})
#将预测结果存到文件中
y_1['price']=np.expm1(y_1['price'])
y_1.to_csv('./ans2_2.csv')
```

问题三的源程序：

```
import scipy.stats as st
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from pylab import mpl
import numpy as np
mpl.rcParams['font.sans-serif'] = ['SimHei'] # 指定默认字体
mpl.rcParams['axes.unicode_minus'] = False
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from tqdm import tqdm
tqdm.pandas()
df1 = pd.read_csv('./data/sale/file1.csv')
df4 = pd.read_csv('./data/sale/file4.csv')
df_11 = df1[['carid','price']]
```

```
df4 = df_11.merge(df4,on='carid')
#取成交的数据
df_trans = df4[df4.withdrawDate.notna()]
df_trans.shape
# # 计算交易周期
df_trans['pushDate'] = pd.to_datetime(df_trans['pushDate'])
df_trans['withdrawDate'] = pd.to_datetime(df_trans['withdrawDate'])
trans_circle = pd.DataFrame(df_trans['withdrawDate'] -
df_trans['pushDate'])
df_trans['transcycle'] = trans_circle[0]
# 转为整型
sep = df_trans['transcycle'].astype(str).str.split(' ', expand=True)
df_trans['transcycle'] = sep[0].astype(int)
df4['transcycle'] = sep[0].astype(int)
# 取 file4 中与 file1 中相同 cardid 的数据
trans_circle_info = df1[df1.carid.isin(df4.carid.tolist())]
# trans_circle_info.info()
df5 = df4[['carid','transcycle']]
NEW_trans_circle = trans_circle_info.merge(df5,on='carid')
NEW_trans_circle
#交易成功的 trans_category 记为 1，交易失败记为 0
NEW_trans_circle.loc[NEW_trans_circle[NEW_trans_circle['transcycle']>
0].index, 'trans_category'] = 1
NEW_trans_circle['trans_category'] =
NEW_trans_circle['trans_category'].fillna(0)
# 缺失值处理，以众数填充
NEW_trans_circle['carCode'] = NEW_trans_circle['carCode'].fillna(1)
NEW_trans_circle['maketype'] = NEW_trans_circle['maketype'].fillna(2)
cols = ["carid", "tradeTime", "brand", "serial", "model", "mileage",
"color", "cityId", "carCode", "transferCount", "seatings",
"registerDate",
       "licenseDate", "country", "maketype", "modelyear",
```

```
"displacement",        "gearbox",        "oiltype",        "newprice",
"price","trans_category"]
df = NEW_trans_circle[cols]
# 以下分类特征全部填充众数
df['carCode'] = df['carCode'].fillna(1)
df['modelyear'] = df['modelyear'].fillna(2017)
df['country'] = df['country'].fillna(779412)
df['maketype'] = df['maketype'].fillna(2)
df['gearbox'] = df['gearbox'].fillna(3)
# # 时间处理(提取年月日)
df['tradeTime'] = pd.to_datetime(df['tradeTime'])
df['registerDate'] = pd.to_datetime(df['registerDate'])
df['licenseDate'] = pd.to_datetime(df['licenseDate'])


df['tradeTime_year'] = df['tradeTime'].dt.year
df['tradeTime_month'] = df['tradeTime'].dt.month
df['tradeTime_day'] = df['tradeTime'].dt.day

df['registerDate_year'] = df['registerDate'].dt.year
df['registerDate_month'] = df['registerDate'].dt.month
df['registerDate_day'] = df['registerDate'].dt.day

df['licenseDate_year'] = df['licenseDate'].dt.year
df['licenseDate_month'] = df['licenseDate'].dt.month
df['licenseDate_day'] = df['licenseDate'].dt.day

del df['tradeTime']
del df['registerDate']
del df['licenseDate']

y_p = df[df['price'] <= 200]
```

```
## 3) 查看预测值的具体频数
plt.hist(y_p['price'], orientation='vertical',
         histtype='bar', color='red')
plt.savefig('./data/sale/price_bar1.png')
plt.show()
df.loc[df[df['price']<= 25].index, 'car_category'] = 0 # 低端车
df.loc[df[(df['price']> 25)&(df['price']<= 50)].index, 'car_category']
= 1 # 中端车
df.loc[df[df['price']> 50].index, 'car_category'] = 2 # 高端车
df_a = df.copy()
df_a['price'] = np.log1p(df_a['price'])
df_a['newprice'] = np.log1p(df_a['newprice'])
df_a
cols = list(df)
cols.insert(0,cols.pop(cols.index('trans_category')))   #   0   是 将
cols.index('price')放在哪一列，cols.pop(cols.index('price')) 是要换的
price 列
df_a = df_a.loc[:,cols] # 开始按照两列互换
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA

# 只举例一个数据
df1 = df_a

data1 = np.array(df1)
# 剩余 4 个 data 略
# 。。。
pred0 = list(df1['car_category'])

pca = PCA(n_components=2)  # 输出两维
newData1 = pca.fit_transform(data1)  # 载入 N 维
```

```
x1, y1 = [], []
x2, y2 = [], []
x3, y3 = [], []
for index, value in enumerate(pred0):
    if value == 0:
        x1.append(newData1[index][0])
        y1.append(newData1[index][1])
    elif value == 1:
        x2.append(newData1[index][0])
        y2.append(newData1[index][1])
    elif value == 2:
        x3.append(newData1[index][0])
        y3.append(newData1[index][1])
plt.figure(figsize=(10, 10))
plt.scatter(x1, y1)
plt.scatter(x2, y2)
plt.scatter(x3, y3)
plt.xticks(fontsize=12, fontweight='bold')  # 默认字体大小为10
plt.yticks(fontsize=12, fontweight='bold')
plt.savefig('./data/sale/第一周卖出的数据分布.png')
plt.show()

from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier
df1=df_a
#转换类型，以低端车为例
data2 = df1[df1['car_category']==0]
data1=pd.DataFrame(data2)
#划分训练集和测试集
x, y = data1.iloc[:, 1:].values, data1.iloc[:, 0].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
```

```
0.2, random_state = 0)
feat_labels = data1.columns[1:]
#标准化
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
#X_test = sc.fit_transform(x_test)
#训练模型
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier

forest =
RandomForestClassifier(n_estimators=300,random_state=42,max_features=
6,n_jobs=-1)
forest.fit(X_train, y_train)
y_pred = forest.predict(X_test)
importances = forest.feature_importances_
indices = np.argsort(importances)[::-1]
for f in range(x_train.shape[1]):
    print("%2d) %-*s %f" % (f + 1, 30, feat_labels[indices[f]],
importances[indices[f]]))
x=sc.transform(x)
y_pred = forest.predict(x)

from sklearn.metrics import f1_score, roc_auc_score, log_loss

print('ROC:', roc_auc_score(y, y_pred))
print('LogLoss:', log_loss(y, y_pred))
```