| | |
|---|---|
| 队伍编号 | MCB2101661 |
| 赛道 | A |

# 基于 Boost 树模型的二手车交易价格评估


## 摘 要

考虑到国内车辆保有量不断增长，二手车交易越变得越来越频繁，对二手车进行合适的估计，能在获得特定利润的前提下，加速车辆的交易周期。

本文对附件 1 和附件 2 中的数据缺失值进行了分析，针对不同类型的特征，分别采用不同的缺失值处理方法，此外，对于数据中存在的异常值，采用箱型图来检测并进行去除，考虑到原始交易价格分布不符合正态分布，对回归预测模型的训练会产生影响，所以使用右偏 np.log1p 函数来进行处理。最终使用 LightGBM 算法来构建交易价格预测模型，所得模型的 $R^2$ 可达 0.9773，MAE 可达 0.07，问题给定的模型性能指标为：0.83597。

对于问题二，本文对附件 4 进行特征重构，得到近 10 维的新特征，包括“降价比”等，同样使用箱型图进行异常值检测与处理，并使用 np.log1p 对数函数进行正太分布处理。最后使用 XGBoost 算法来进行模型构建，其训练集为短期内交易完成的车辆数据，测试集为短期内未完成交易的车辆数据（因变量仍为交易价格）。最终所得模型的 $R^2$ 为 0.88424，MAE 为 0.18171。所训练的模型，可以用于对短期内未交易的车辆进行价格调整预测，有助于提高在库车辆的交易速度。



关键词：二手车估价，机器学习，XGBoost，Lightgbm

目录

# 1 问题介绍

## 1.1 研究背景

随着我国经济的飞速发展，国内居民机动车保有量不断增长，据国家统计局发布的《2018 年国民经济和社会发展统计公报》显示，我国私人汽车保有量首次突破 2 亿辆，达到 2.07 亿辆，增长 10.9%，私人轿车接近 1.26 亿辆，增长 10.3%[1]。相应的，现如今二手车交易也随之逐步发展，越来越多的机动车以“二手车”形式在流通环节，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，究其原因是二手车价格难于准确估计和设定，不但受到车本身基础配置，如品牌、车系、动力等的影响，还受到车况如行驶里程、车身受损和维修情况等的影响，甚至新车价格的变化也会对二手车价格带来作用。

## 1.2 研究问题

基于以上背景，需要通过数据分析与建模的方法解决下面的问题：

**（1）问题一：针对零售场景的估价问题**

基于给定的二手车交易样本数据（附件 1：估价训练数据），选用合适的估价方法，构建模型，预测二手车的零售交易价格。

**（2）问题二：加速在库车辆销售速度**

假设你们是门店的定价师，请你们结合附件 4“门店交易训练数据”对车辆的成交周期（从车辆上架到成交的时间长度，单位：天）进行分析，挖掘影响车辆成交周期的关键因素，加快在库车辆的销售速度。

**（3）问题三：**

依据给出的样本数据集，思考还有哪些值得研究的问题，并给出思路。

# 2 数据预处理

## 2.1 问题一数据预处理

### 2.1.1 缺失值占比分析

原始数据（包括附件 1 与附件 2）的缺失值占如表 1 所示，通过对比分析得出一下结论：

1. 训练数据与验证数据中匿名特征 4、7、15 缺失值占比均超过了 35%，删除该维度。
2. 其余缺失特征需要进一步探索其与数据之间的关系来考虑数据清洗的方法（见 2.3）。
3. 训练数据中不包含缺失值的特征共计 22 维，每一维 30000 行数据。验证数据中不包含缺失值的特征也共计 22 维，但是二者的特征种类并不一致。

表 1 附件 1 与附件 2 数据中的缺失值占比

| 训练数据 | | | | 验证数据 | | | |
|---|---|---|---|---|---|---|---|
| 特征名称 | 缺失值占比 | 特征名称 | 缺失值占比 | 特征名称 | 缺失值占比 | 特征名称 | 缺失值占比 |
| 车辆 id | 0.00% | 燃油类型 | 0.00% | 车辆 id | 0.00% | 燃油类型 | 0.00% |
| 展销时间 | 0.00% | 新车价 | 0.00% | 展销时间 | 0.00% | 新车价 | 0.00% |
| 品牌 id | 0.00% | 匿名特征 1 | 5.27% | 品牌 id | 0.00% | 匿名特征 1 | 6.80% |
| 车系 id | 0.00% | 匿名特征 2 | 0.00% | 车系 id | 0.00% | 匿名特征 2 | 0.00% |
| 车型 id | 0.00% | 匿名特征 3 | 0.00% | 车型 id | 0.00% | 匿名特征 3 | 0.00% |
| 里程 | 0.00% | 匿名特征 4 | 40.36% | 里程 | 0.00% | 匿名特征 4 | 37.26% |
| 车辆颜色 | 0.00% | 匿名特征 5 | 0.00% | 车辆颜色 | 0.00% | 匿名特征 5 | 0.00% |
| 车辆所在城市 id | 0.00% | 匿名特征 6 | 0.00% | 车辆所在城市 id | 0.00% | 匿名特征 6 | 0.00% |
| 国标码 | 0.03% | 匿名特征 7 | 60.15% | 国标码 | 0.00% | 匿名特征 7 | 66.30% |
| 过户次数 | 0.00% | 匿名特征 8 | 12.58% | 过户次数 | 0.00% | 匿名特征 8 | 8.32% |
| 载客人数 | 0.00% | 匿名特征 9 | 12.48% | 载客人数 | 0.00% | 匿名特征 9 | 8.26% |
| 注册日期 | 0.00% | 匿名特征 10 | 20.80% | 注册日期 | 0.00% | 匿名特征 10 | 24.62% |
| 上牌日期 | 0.00% | 匿名特征 11 | 1.54% | 上牌日期 | 0.00% | 匿名特征 11 | 1.46% |
| 国别 | 12.52% | 匿名特征 12 | 0.00% | 国别 | 7.92% | 匿名特征 12 | 0.02% |
| 厂商类型 | 12.14% | 匿名特征 13 | 5.40% | 厂商类型 | 7.50% | 匿名特征 13 | 5.20% |
| 年款 | 1.04% | 匿名特征 14 | 0.00% | 年款 | 2.12% | 匿名特征 14 | 0.00% |
| 排量 | 0.00% | 匿名特征 15 | 91.93% | 排量 | 0.00% | 匿名特征 15 | 94.38% |
| 变速箱 | 0.00% | 交易价格 | 0.00% | 变速箱 | 0.00% | | |

### 2.1.2 缺失数据处理

去除缺失值占比较大的特征维度之后，余下的包含缺失值的特征表如下所示（四舍五入）。

从表中我们可以发现，训练集与测试集之间的缺失值占比类型大致相同，主要是：“国别、厂商类型、年款、匿名特征 1，8，9，10，11，13”。其余的如“国标码，变速箱”等由于缺失值占比较小，且验证集中没有缺失，所以建议：将训练数据中的“国标码”、“变速箱”特征下的缺失值进行“行删除”，删除“匿名特征 12”维度列。

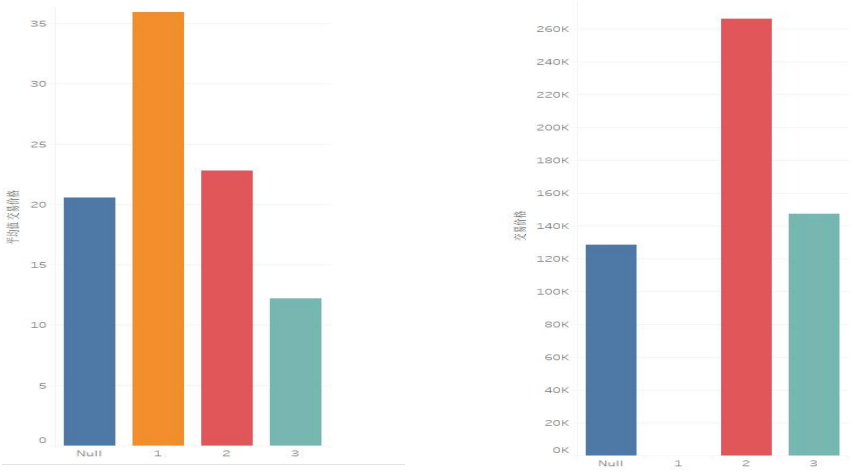
余下含有缺失值的特征，按照缺失占比大小进行排序，逐个进行分析

表 2 按照缺失值占比对附件 1 和附件 2 中的特征进行排序

| 训练数据 | | 验证数据 | |
|---|---|---|---|
| 特征名称 | 缺失值占比 | 特征名称 | 缺失值占比 |
| 国标码 | 0.03% | | |
| **国别** | **12.52%** | **国别** | **7.92%** |
| **厂商类型** | **12.14%** | **厂商类型** | **7.50%** |
| **年款** | **1.04%** | **年款** | **2.12%** |
| 变速箱 | 0.00% | 匿名特征 12 | 0. |
| **匿名特征 1** | **5.27%** | **匿名特征 1** | **6.80%** |
| **匿名特征 8** | **12.58%** | **匿名特征 8** | **8.32%** |
| **匿名特征 9** | **12.48%** | **匿名特征 9** | **8.26%** |
| **匿名特征 10** | **20.80%** | **匿名特征 10** | **24.62%** |
| **匿名特征 11** | **1.54%** | **匿名特征 11** | **1.46%** |
| **匿名特征 13** | **5.40%** | **匿名特征 13** | **5.20%** |

**（1）匿名特征 10**

匿名特征 10 属于分类变量，包含 Null 缺失值在内一共是 4 个类别，各类之间与 price（交易价格）之间的关系如下图所示，使用 Tableau 进行制图，略有不清晰：




（a）每类中的平均交易价格　　（b）总计的交易价格

图 1 匿名特征 10 的交易价格分布图

因此，对于“匿名特征 10”，建议将 Null 缺失值重构为类别 4.

**（2）匿名特征 8**

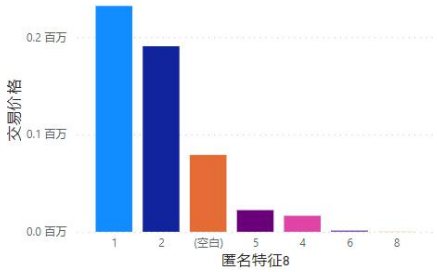



图 2 匿名特征 8 的交易价格分布

匿名特征 8 的交易价格分布如图 2 所示，同理，将匿名特征 8 的 NULL 缺失值重构为类别 3；

**（3）国别**

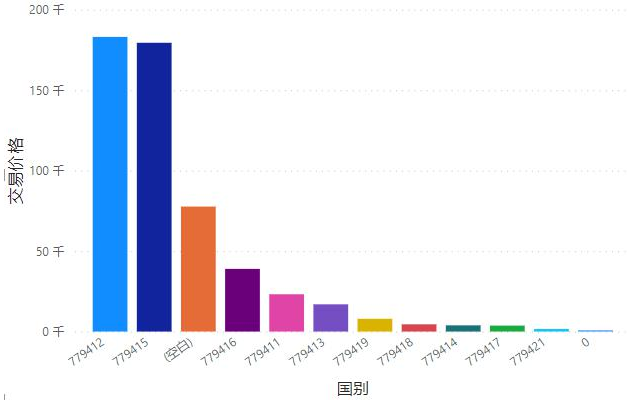



图 3 不同国别的交易价格分布图

如图 3 所示，将国别进行排序，共得到 12 类数字。将 NULL 缺失值重构为类别 773420，将类别 0 重命名为 773410

**（4）匿名特征 9**

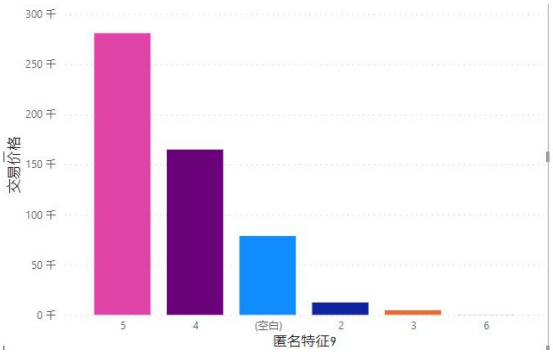



图 4 匿名特征 9 的交易价格分布图

同理分析，如图 4 所示，将 NULL 缺失值重构为类别 1

**（5）厂商内型**

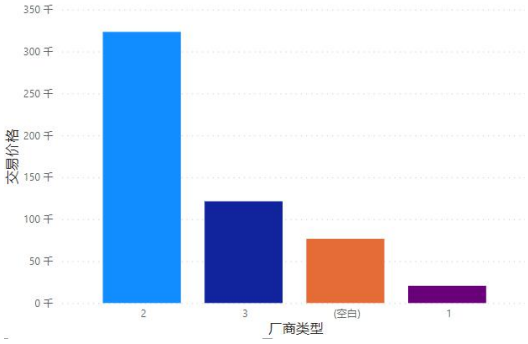



图 5 不同厂商内省的交易价格分布图

同理，如图 5 所示，将 NULL 缺失值重构为类别 4

**（6）匿名特征 13**

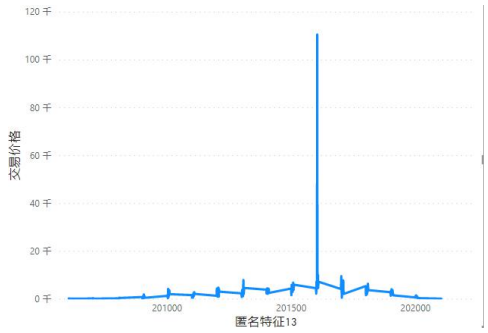



图 6 匿名特征 13 的交易价格分布图

如图 6 所示，“匿名特征 13”包含 NULL 值在内一共有 168 维度类别，由于类别数量较大，所以我们无法进行简单的填补，此外通过观察匿名特征 13 的数据分布，我猜测特征是与时间有关的类别，类似最终你的交易日期什么的。除了某一个月交易额激增，其余无显著差异。综合分析，最好删除该列

**（7）匿名特征 1**

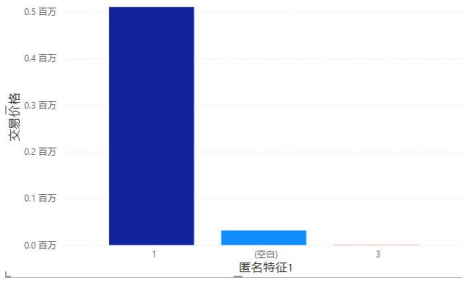
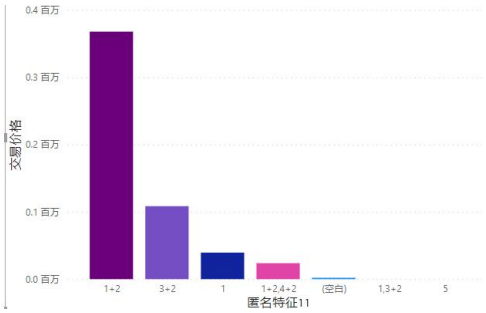



图 7 匿名特征 1 的交易价格分布图

“匿名特征”1 为分类变量，通过图表分析，建议将 NULL 缺失值重构为类别 2

**（8）匿名特征 11**




```
In  [222]:  1  data_1["匿名特征11"].unique()
Out[222]:  array(['1', '1+2', nan, '3+2', '1+2,4+2', '1,3+2', '5'], dtype=object)

In  [221]:  1  data_2["匿名特征11"].unique()
Out[221]:  array(['1+2', nan, '1+2,4+2', '1', '3+2'], dtype=object)
```

图 8 匿名特征 11 的交易价格分布图

如图 8 所示，考虑到训练数据与测试数据之间分类缺失的类型无法确定，所以建议删除该列特征

**（9）年款**

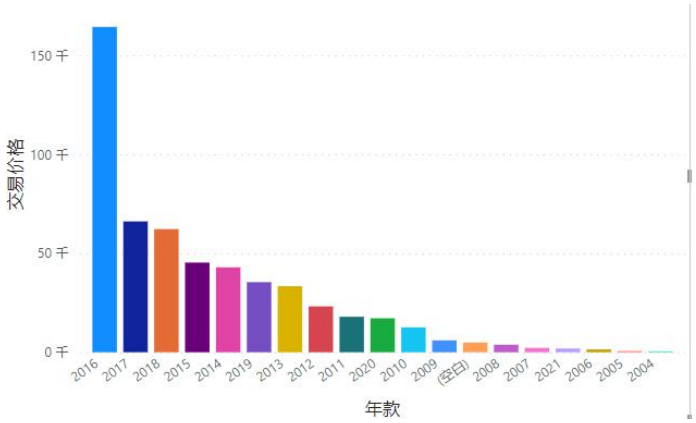



图 9 不同年款的交易价格分布图

“年款”特征表示的是该车是哪年出的，通过分析，建议将缺失值 NULL 重构为类别 2003.

到此，训练数据与验证数据中的缺失值均被处理。

## 2.1.3 异常值处理

如图 10 所示，使用箱型图进行异常值检测，计算上四分位数、中位数、下四分位 数以及均值 (四分位数间距 IQR = Q3−Q1)。将大于上四分位数 3 倍四分位数差 的值，或者小于下四分位数 3 倍四分位数差的值定义为异常值（也就是常说的极端异常值）。图 11 为，使用箱型图处理的附件 1 中的交易价格。

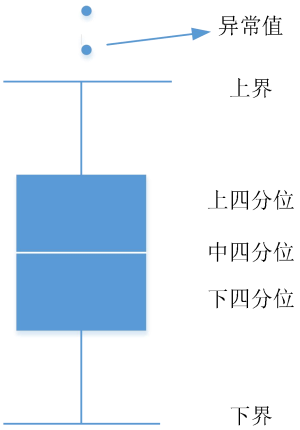



图 10 使用箱型图检测异常值

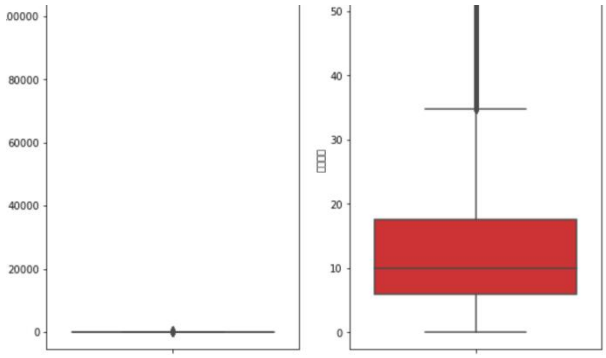



图 11 问题一交易价格异常值处理

最终所得，可用于模型训练的数据为 29355 行。

## 2.2 问题二数据预处理

### 2.2.1 缺失值占比

附件 4 中的缺失值占比如下：

表 3 附件 4 中的缺失值占比

| 特征名称 | 数据类型 |
|---|---|
| 车辆 id | 0.00% |
| 上架时间 | 0.00% |
| 上架价格 | 0.00% |
| 价格调整 | 0.00% |
| 下架时间 | 0.00% |
| 成交时间 | 20.00% |

### 2.2.2 缺失数据处理

题目中提到“附件 4“门店交易训练数据”包括 6 个字段，如下表所示，其中所 有 carid 等相关信息包含在附件 1“估价训练数据”中。”

通过车辆 ID 将附件 1 与附件 4 进行拼接，附件 1 的数据预处理如上文所述。本文中认为：附件 1 中存在交易价格的车辆一定完成了交易，所以，结合下架时间（成交车辆下架时间和成交时间相同），对成交时间的缺失值进行填充。即：**使用拼接的数表来对车辆的成交时间进行缺失值的填补，并以此计算出“成交周期”**

### 2.2.3 异常值处理

使用箱型图进行异常值处理，详情可见上文。图 12 为问题二中交易周期的异常值处理示意图。

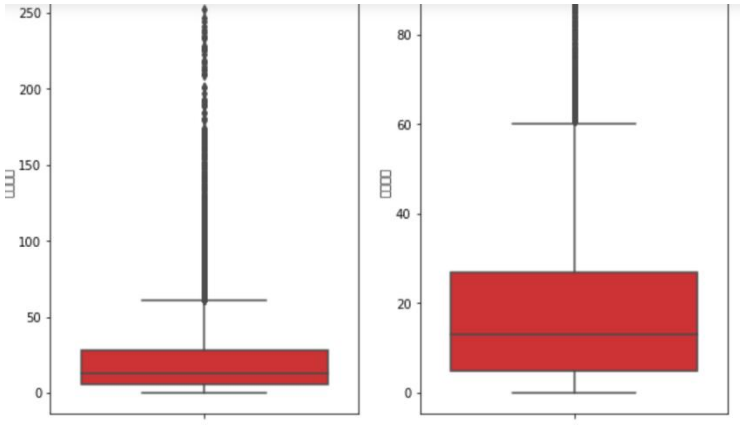



图 12 附件 4 中交易周期的异常值处理

最终所得可用于模型构建的数据数量为 9535 行。

# 3 模型构建

## 3.1 问题一

问题一的核心就是根据附件一给定的数据构建合适的二手车价格预测模型，然后使用建立的预测模型对附件二中的数据进行预测，并将结果提交给附件三中。

模型的评价指标如公式（1）所示，涉及了平均相对误差和误差准确率。

$$0.2 * （1 - \text{Mape}） + 0.8 * \text{Accuracy}_5 \quad （1）$$

附件 1 中包含有 15 维匿名特征，总共有 36 维特征，各指标的重要性并未告知。

本文使用的预测模型为机器学习模型，考虑到预测回归模型有很多，如：线性回归、岭回归、支持向量机回归等等，通过查阅相关论文资料，本文使用 LightGBM 算法来进行模型构建，属于 Boost 集成算法的一种[2]。问题一的基础流程如图 1 所示。

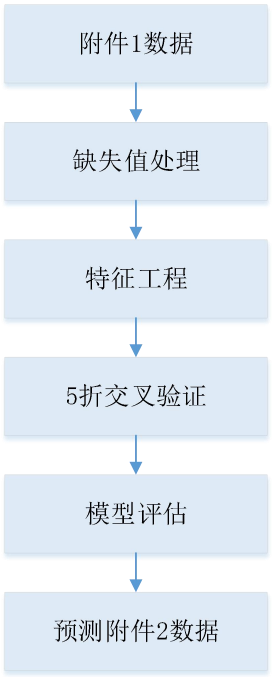



图 1 问题一模型的构建基础流程图

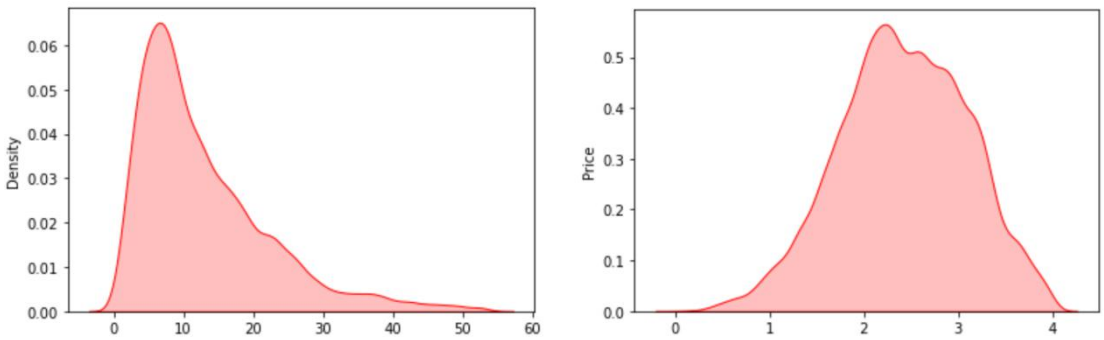
### 3.1.1 特征处理

（1）提取时间特征

现将“展销时间”、“注册日期”、“上牌日期”的数据格式转换为 pandas 的 Datetime[64]格式，然后基于此，提取对应的“年”、“月”、“日”特征。

（2）查看标签分布

原始标签的数据分布，如图 1（a）所示，数据分布并不符合正态分布，是右偏数据，回归中对数据分布较为敏感，如果不符合正态分布需要进行数据转换成近似正态分布，使用使用对数的右偏变换函数，将数据分布转为近似正态分布(np.log1p)。




（a）原始交易价格数据分布图　　（b）对数变换后交易价格的数据分布图

图 1 交易价格的数据分布图

（3）特征无量纲化

本文使用数据标准化来进行无量纲化，将数据(x)按均值(μ)中心化后，再按标准差(σ)

缩放，其计算公式如公式（2）所示。

$$x^* = \frac{x - \mu}{\sigma}$$

### 3.1.2 模型训练

对于问题一，使用 5 折交叉验证，结合 LightGBM 算法进行模型训练，参数调节使用穷举法，最后模型的学习率设置为 0.01，损失函数使用 MAE，最大深度为 8，叶子数为 32 等。

### 3.1.3 模型评价

针对回归预测模型，常用的指标有 R2 和 MAE，此外，题目中还给定了相应的评价指标如公式（1）所示。

最终对于问题一，我们得到模型训练效果如下所示：

| | |
|---|---|
| R2 | 0.9773333233690317 |
| MAE | 0.07008888321423438 |
| 题目给定的评价指标 | 0.83597 |

## 3.2 问题二

估价问题解决后，在所得估价基础上，需要多久能够完成交易也就成了二手车交易需要关注的问题。问题二给定了附件 4，包括了车辆 ID、上架时间等字段，可以通过车辆 ID 将附件 4 与附件 1 进行联结，进行进一步分析与数据挖掘。

主要问题是：你可以采取哪些因素来加快在库车辆销售速度？。由于题目中表明“假设你作为门店的定价师”，所以我们所能更改的因素主要还是二手车价格。

因此对于问题二，将数据根据交易周期的大小进行分箱，分为了 6 类，包括 1 周内交易、两周内交易、三周内交易、四周内交易、五周内交易、五周外交易。通过对不同交易周期的数据进行 AB 测试（方差分析），我们可以得到，不同交易周期间，没有显著差异影响的特征量，将相应特征进行过滤。

在处理完冗余特征以及数据缺失值后，构建 XGBoost 回归模型[3]，其中训练集为前 5 周内完成交易的数据，测试集为 5 周外交易的数据（因变量为交易价格）。通过使用所构建的模型来对附件 4 中交易周期过长的车辆价格进行回归预测，可以得到有望在较短时间内完成车辆交易的最新定价。

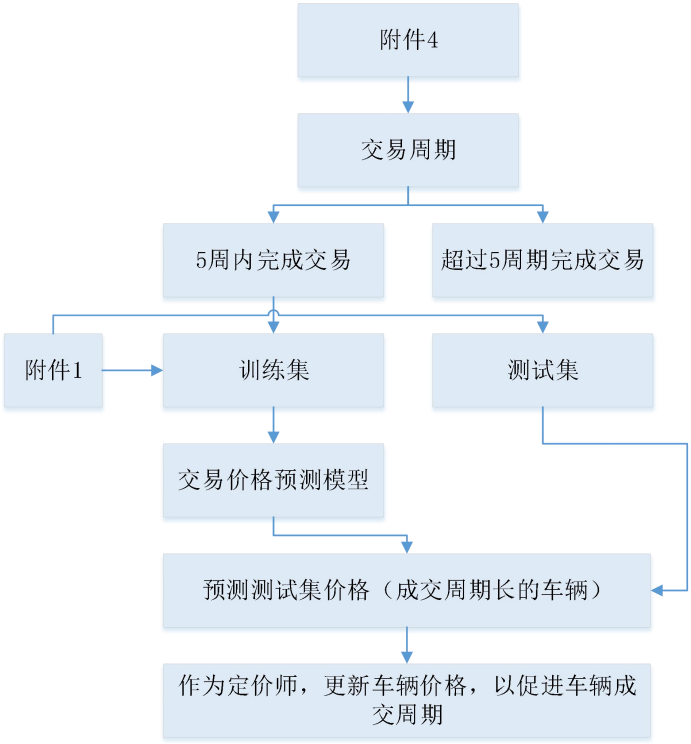



图 2 问题二的技术路线流程图

### 3.2.1 特征处理

（1）结合专家知识，将附件 4 的内容与附件 1 拼接。并对“价格调整”栏进行维度重构，提取出 5 维新的特征，如下：“是否调价”“调价次数”“调价频率（次/天）”“最终调价时间”“最终调价”。

此外，考虑到二手车交易受到价格的影响较大，所以对比新车价等因素，最终对“初始降价”（二手车第一次定价与新车价的差值）、“最终降价”（二手车交易价格与新车价的差值）、“最终降价比”等。

（2）将数据根据交易周期的大小进行分箱，分为了 6 类，包括 1 周内交易、两周内交易、三周内交易、四周内交易、五周内交易、五周外交易。通过对不同交易周期的数据进行 AB 测试（方差分析），去除对于交易周期影响不大的特征，如：’车辆 id’，’展销时间’，’车辆颜色’，’车辆所在城市 id’等，共计 19 维特征。

（3）其余步骤，同 3.1.1

### 3.2.2 模型构建

车辆能否成功交易，除了取决于销售的谈判技巧，更重要的是车辆本身是否受消费者青睐，价格是否公道。作为定价师，我们可以操作的最有效的手段就是价格，所以针对问题二的数据，使用 XGBoost 算法进行模型构建（因变量为交易价格），只不过仅训练短期内进行交易的价格预测模型，并用其对短期内未进行交易的车辆重新进行价格预

测，将预测的价格作为相应车的调整价格，已达到加速在库车辆交易速度的目的。

### 3.2.3 模型评估

使用 R2 和 MAE 作为模型的评估指标，将短期内未进行交易的车辆作为训练，进行预测，模型的性能表现如下所示，其价格预测值与现有车辆价格对比如图 4 所示。

| | |
|---|---|
| 测试集 R2 | 0.884243029 |
| 测试集 MAE | 0.181712527 |

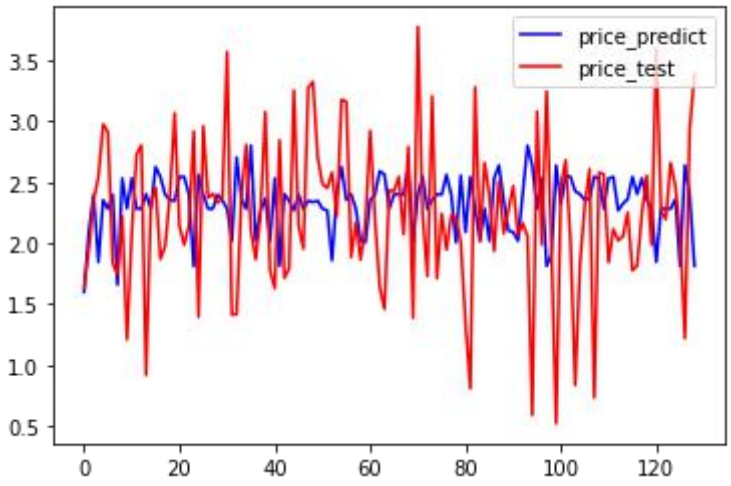



图 4 价格预测值与车辆现价对比图

## 2.3 问题三

### 2.3.1 问题描述

可以值得研究的问题：对于二手车商家来说，应采购按照什么标准来采购车辆？

通过问题一我们分析了影响最终交易价格的重要特征，通过问题二，我们分析了哪些因素可以缩短交易周期。因此，综合上述的特征，商家在采购二手车时，就可以利用上面所挖掘一些的重要特征去采购二手车。比如具有某些特种的车辆，其交易价格较高从而利润也较大，同时成交周期也相对较短，就可以多采购一些。而某些车辆明显难以通过特征评判，或者明显难以交易，或者交易后利润低，就应该少采购。从而可以使商家尽可能获取更多的利润。

### 2.3.2 思路

（1）二手车采购的评价指标

首先我们可以通过某款车最终的平均利润 p 和平均成交周期 t 等因素来定义一个评价指标 S 来衡量采购的二手车的价值，S 与 p 成正比，与 t 成反比。然后再来考虑哪些

因素会对 S 造成直接且重要的影响。

（2）可能的影响因素

考虑在采购二手车时，哪些因素可能会对评价指标 S 造成影响。

一方面，可以直接利用已经给出的数据特征，如品牌、车型、里程、新车价等进行分析。

另一方面，由于不知道匿名特征的具体含义，可以考虑添加一些的特征数据：如是否改装（改装次数）、整体车况（维修次数、是否为事故车）、维保记录、车辆用途、油耗、车辆保险的影响、二手车的售后保证。

是否改装（改装次数）：对于改装的车辆，车主是认为应该在原本的车价上增加价格的，但车辆的改装会影响原来的车况，并且带有很强的个人爱好趋向，买家可能并不愿意为不喜欢的改造买单，所以该项特征也及其重要。

整体车况（维修次数）：整体的车况即车的性能状况，肯定是买家所关心的重要因素之一，对于发生过事故，有多次维修历史，车况明显下降的车辆，买家也是不愿接受的。

维保记录：只要车辆进到 4S 店进行检修，就会留下记录，包括查询车辆的出险记录。卖家对于车辆的保养状况，车主对与车辆保养的越好，维保记录越详细，买家就更容易为其买单。

车辆用途：车辆用途，主要是看行驶证上的使用性质，非营运、营转非、运营等等，用途还包括是否有跑黑车、公司业务用车等（一般公里数都比较高），知道前车主的使用性质，对车辆价格有更好的把握。

使用成本：重点关心的是油耗与维护成本。排量高油耗车型保值度较低，维修网点多寡与配件价格的高低对成交价格有影响，维修网点越多，价格相对较低的车型，其后期的维修保养费用就会越低，进而这样的车型也就越受欢迎，买来之后也就越划算。

车辆保险：车辆的保险的种类、数量，关心到买家与卖家的直接的经济利益，买家是否愿意为这些保险买单。

二手车的售后：二手车有无售后、售后是否完善，是买家在平台购买二手车重点考虑的因素。

在考虑以上一些可能的影响因素之后，在有数据的情况下，可以对其进行建模。

由于目前缺乏采购二手车的数据，暂时无法进行具体地建模和分析，因此各方面是否考虑全面，以及此思路是否具有可行性无法进行判断，此处仅提出一种思路。

# 参考文献


[1] 国家统计局.2018 年国家经济和社会发展统计公报[EB/OL]. http://www. stats. gov.cn/tjsj/zxfb/201902/t20190228_1651265.html

[2] Meng Q . LightGBM: A Highly Efficient Gradient Boosting Decision Tree. 2018.

[3] Chen T , Guestrin C . XGBoost: A Scalable Tree Boosting System[C]// the 22nd ACM SIGKDD International Conference. ACM, 2016.

[4] Liang, Weizhang et al. “Predicting Hard Rock Pillar Stability Using GBDT, XGBoost, and LightGBM Algorithms.” (2020).

[5] Massaoudi, Mohamed et al. “A novel stacked generalization ensemble-based hybrid LGBM-XGB-MLP model for Short-Term Load Forecasting.” Energy (2021): n. pag.

[6] 陈洞明. 基于数学模型的二手车快速估价方法的研究[J]. 时代汽车,2019(18):144-146. DOI:10.3969/j.issn.1672-9668.2019.18.065.

[7] 王静娜. 基于随机森林算法的二手车估价模型研究[D]. 2019.

[8] 曹睿,廖彬,李敏,等. 基于 XGBoost 的在线短租市场价格预测及特征分析模型[J]. 数据分析与知识发现,2021(6). DOI:10.11925/infotech.2096-3467.2020.1186.

[9] 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). DOI:10.3969/j.issn.1000-8519.2021.21.016.

[10] Los, Helena et al. “Evaluation of Xgboost and Lgbm Performance in Tree Species Classification with Sentinel-2 Data.” 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS (2021): 5803-5806.

[11] 王冬雪,郭秀娟. 基于 XGBoost 算法的房价预测模型[J]. 北方建筑,2021(3). DOI:10.3969/j.issn.2096-2118.2021.03.023.

[12] Omar, Kamil Belkhayat Abou. “XGBoost and LGBM for Porto Seguro ’ s Kaggle challenge : A comparison Semester Project.” (2018).

[13] 田秋红,廖文琪,欧阳汉. 基于 XGBoost 的大宗商品价格预测[J]. 企业科技与发展,2021(4). DOI:10.3969/j.issn.1674-0688.2021.04.054.

[14] 国网浙江省电力有限公司. 一种基于 LSTM 与 LGBM 的电力负荷预测方法:CN201910338745.1[P].

# 附件 1 问题一数据清洗

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")


data_1 = pd.read_csv("./附件/附件 1：估价训练数据.txt",sep="\t",header=None)
data_2 = pd.read_csv("./附件/附件 2：估价验证数据.txt",sep="\t",header=None)

table_1 = pd.read_excel(r"C:/Users/Administrator/Desktop/附件 1：维度表.xlsx")
column1 = dict(table_1["Description"][:-2])
column2 = {}
for i in range(1,16,1):
    column2[i+19] = "匿名特征{a}".format(a=i)

data_1.rename(columns=column1,inplace=True)
data_1.rename(columns=column2,inplace=True)
data_2.rename(columns=column1,inplace=True)
data_2.rename(columns=column2,inplace=True)

data_1.rename(columns={35:"交易价格"},inplace=True)

import os
Floder_Path = "./数据/"
if not os.path.exists(Floder_Path):
    os.mkdir(Floder_Path)
Save_Path = Floder_Path + "train_data_raw.xlsx"
data_1.to_excel(Save_Path,index=False)
data_2.to_excel(Floder_Path+"val_data_raw.xlsx")

data_1["展销时间"] = pd.to_datetime(data_1["展销时间"],format="%Y-%m-%d")
data_1["注册日期"] = data_1["注册日期"].astype("datetime64")
```

```
data_1["上牌日期"] = data_1["上牌日期"].astype("datetime64")

# 没有缺失值的特征
data_1.iloc[:,data_1.isnull().mean().values==0].to_excel("./数据/train_data_notnull.xlsx")
data_2.iloc[:,data_2.isnull().mean().values==0].to_excel("./数据/val_data_notnull.xlsx")

data_1["匿名特征 10"].fillna(4,inplace=True)
data_1["匿名特征 8"].fillna(3,inplace=True)
data_1["国别"].fillna(773420,inplace=True)
data_1["国别"].replace(0,773410,inplace=True)
data_1["匿名特征 9"].fillna(1,inplace=True)
data_1["厂商类型"].fillna(4,inplace=True)
data_1.drop(["匿名特征 13","匿名特征 11","匿名特征 12","匿名特征 4","匿名特征 7","匿名特征 15"],axis=1,inplace=True)
data_1["匿名特征 1"].fillna(2,inplace=True)
data_1["年款"].fillna(2003,inplace=True)
data_1.dropna(axis=0,how = "any",inplace = True )

Save_Path = Floder_Path + "train_data_new.xlsx"
data_1.to_excel(Save_Path,index=False)

data_2["匿名特征 10"].fillna(4,inplace=True)
data_2["匿名特征 8"].fillna(3,inplace=True)
data_2["国别"].fillna(773420,inplace=True)
data_2["匿名特征 9"].fillna(1,inplace=True)
data_2["厂商类型"].fillna(4,inplace=True)
data_2.drop(["匿名特征 13","匿名特征 12","匿名特征 11","匿名特征 4","匿名特征 7","匿名特征 15"],axis=1,inplace=True)
data_2["匿名特征 1"].fillna(2,inplace=True)
data_2["年款"].fillna(2003,inplace=True)

Save_Path = Floder_Path + "val_data_new.xlsx"
data_2.to_excel(Save_Path,index=False)
```

# 附件 2 问题一模型构建

```
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 import metrics
import pandas as pd
import warnings
from sklearn.preprocessing import scale
from sklearn.model_selection import cross_val_score
import lightgbm as lgb
import time
import numpy as np
from sklearn.preprocessing import StandardScaler



train = pd.read_excel(r"./数据/train_data_new.xlsx")
test = pd.read_excel("./数据/val_data_new.xlsx")

train["展销时间"] = pd.to_datetime(train["展销时间"])
train["注册日期"] = pd.to_datetime(train["注册日期"])
train["上牌日期"] = pd.to_datetime(train["上牌日期"])

test["展销时间"] = pd.to_datetime(test["展销时间"])
test["注册日期"] = pd.to_datetime(test["注册日期"])
test["上牌日期"] = pd.to_datetime(test["上牌日期"])



train['展销时间_year'] = train['展销时间'].dt.year
train['展销时间_month'] = train['展销时间'].dt.month
train['展销时间_day'] = train['展销时间'].dt.day
train['注册日期_year'] = train['注册日期'].dt.year
train['注册日期_month'] = train['注册日期'].dt.month
```

```
train['注册日期_day'] = train['注册日期'].dt.day
test['展销时间_year'] = test['展销时间'].dt.year
test['展销时间_month'] = test['展销时间'].dt.month
test['展销时间_day'] = test['展销时间'].dt.day
test['注册日期_year'] = test['注册日期'].dt.year
test['注册日期_month'] = test['注册日期'].dt.month
test['注册日期_day'] = test['注册日期'].dt.day
del train['展销时间']
del test['展销时间']
del train['注册日期']
del test['注册日期']
del train['上牌日期']
del test['上牌日期']

from Box_Figure import outliers_proc
train_X = outliers_proc(train,'交易价格',scale=3) #  使用箱型图处理异常值
train_X['交易价格'] = np.log1p(train_X['交易价格'])
train_y = train_X['交易价格']
del train_X['交易价格']

y = train_y
fig = sns.kdeplot(y, color='Red', shade=True)
scatter_fig = fig.get_figure()

data = train_X
scaler = StandardScaler()
train_x = scaler.fit_transform(train_X)
test_x = scaler.fit_transform(test)

#  模型的参数
params = {'learning_rate': 0.01,
                'boosting_type': 'gbdt',
                'objective': 'regression_l1',
                'metric': 'mae',
                'min_child_samples': 46,
```

```
                'min_child_weight': 0.01,
                'feature_fraction': 0.8,
                'bagging_fraction': 0.8,
                'bagging_freq': 2,
                'num_leaves': 32,
                'max_depth': 8,
                'n_jobs': -1,
                'seed': 2021,
                'verbose': -1,
                }


    def Accuracy(y_true, y_pred): #赛题给定的指标 MAPE

        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 Accuracy

    val_pred = np.zeros(len(train_x))
    val_true = np.zeros(len(train_x))
    preds = np.zeros(len(test_x))
    folds = 5
    kfold = KFold(n_splits=folds, shuffle=True, random_state=4071)
    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[val_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=(
```

```
                                    train_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

from sklearn import metrics
acc = Accuracy(val_true, val_pred)
print('*'*50)
print('Accuracy ', round(acc, 5))
print("R2:",metrics.r2_score(val_true,val_pred))
print("MAE:",metrics.mean_absolute_error(val_true,val_pred))
```

# 附件 3 问题二数据清洗与特征构造

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import os
import ast

data_1 = pd.read_excel("./数据/train_data_new.xlsx")

data_4 = pd.read_csv("./附件/附件 4：门店交易训练数据.txt",sep="\t",header=None)
column4 = {0:"车辆  id",1:"上架时间",2:"上架价格",3:"价格调整",4:"下架时间",5:"成交时间"}
data_4.rename(columns=column4,inplace=True)

Floder_Path = "./问题二/"
if not os.path.exists(Floder_Path):
    os.mkdir(Floder_Path)
Save_Path = Floder_Path + "data_4.xlsx"
data_4.to_excel(Save_Path,index=False)

data_merge = pd.merge(left=data_1,right=data_4,on="车辆  id")

data_merge = pd.merge(left=data_merge,right=data_4,how="outer")

PA_whether = {}
PA_price = {}
PA_time = {}
PA_num = {}
PA_num = {}
PA_freq = {}
for i in range(data_merge.shape[0]):
    PA = data_merge["价格调整"].values[i]
```

```
        PA = ast.literal_eval(PA)
        if not PA:
            PA_price[i] = data_merge["上架价格"].values[i]
            PA_time[i] = data_merge["上架时间"].values[i]
            PA_num[i] = 0
            PA_freq[i] = 0
            PA_whether[i] = False
        else:
            keys = []
            values = []
            for key,value in PA.items():
                keys.append(key)
                values.append(value)

            PA_price[i] = values[-1]
            PA_time[i] = keys[-1]
            PA_num[i] = len(keys)
            days = np.timedelta64(pd.to_datetime(PA_time[i])-pd.to_datetime(data_merge["上架时间"].values[i]),"D").astype(float)
            PA_freq[i] = PA_num[i]/days
            PA_whether[i] = True


    PA_whether = pd.Series(PA_whether)
    PA_freq = pd.Series(PA_freq)
    PA_time = pd.Series(PA_time)
    PA_price = pd.Series(PA_price)
    PA_num = pd.Series(PA_num)

    data_merge.insert(data_merge.shape[1]-1,"是否调价",PA_whether)
    data_merge.insert(data_merge.shape[1]-1,"调价次数",PA_num)
    data_merge.insert(data_merge.shape[1]-1,"调价频率",PA_freq)
    data_merge.insert(data_merge.shape[1]-1,"最终调价时间",PA_time)
    data_merge.insert(data_merge.shape[1]-1,"最终调价",PA_price)
```

```
data_merge.insert(data_merge.shape[1],"成交时间 New",data_merge[data_merge["交易价格"].notnull()]["下架时间"])
data_merge["成交时间 New"][-7:] = data_merge["成交时间"][-7:]
data_merge["成交周期"] = data_merge["成交时间New"].astype("datetime64")-data_merge["上架时间"].astype("datetime64")
data_merge["成交周期"] = data_merge["成交周期"].astype('timedelta64[D]').astype(int)



data_merge.dropna(subset=["年款"],inplace=True)

# 成交周期分箱
data_merge["成交速度"] = 1
data_merge["成交速度"][data_merge["成交周期"]<=7] = 1
data_merge["成交速度"][(data_merge["成交周期"]>7)&(data_merge["成交周期"]<=14)] = 2
data_merge["成交速度"][(data_merge["成交周期"]>14)&(data_merge["成交周期"]<=21)] = 3
data_merge["成交速度"][(data_merge["成交周期"]>21)&(data_merge["成交周期"]<=28)] = 4
data_merge["成交速度"][(data_merge["成交周期"]>28)&(data_merge["成交周期"]<=35)] = 5
data_merge["成交速度"][data_merge["成交周期"]>35] = 6

# 交易价格分箱
data_merge["车辆级别"] = 1
data_merge["车辆级别"][(data_merge["交易价格"]>20)&(data_merge["交易价格"]<=50)] = 2
data_merge["车辆级别"][(data_merge["交易价格"]>50)&(data_merge["交易价格"]<=100)] = 3
data_merge["车辆级别"][data_merge["交易价格"]>100] = 4

#降价比
data_merge["初始降价"] = data_merge["新车价"]-data_merge["上架价格"]
data_merge["最终降价"] = data_merge["新车价"]-data_merge["最终调价
```

"].astype(float)

data_merge["初始降价比"] = (data_merge["新车价"]-data_merge["上架价格"])/data_merge["新车价"]

data_merge["最终降价比"] = (data_merge["新车价"]-data_merge["最终调价"].astype(float))/data_merge["新车价"]

```
mid = data_merge["成交周期"]
data_merge.drop(columns=["成交周期"],inplace=True)
data_merge.insert(data_merge.shape[1],"成交周期",mid)

mid = data_merge["成交速度"]
data_merge.drop(columns=["成交速度"],inplace=True)
data_merge.insert(data_merge.shape[1],"成交速度",mid)
```

# 附件 4 问题二模型构建

```
from Box_Figure import outliers_proc    #箱型图
from xgboost import XGBRegressor
from xgboost import XGBRFRegressor
from sklearn.ensemble import RandomForestRegressor
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import os
import ast

data_merge = pd.read_excel("./问题二/data_merge.xlsx")

data_merge.drop(columns="成交时间",inplace=True)

data_merge.drop(columns=["成交时间 New","下架时间"],inplace=True)

for column in ["成交周期","交易价格","最终调价"]:
# column = "成交周期"
    data_merge = outliers_proc(data_merge,column,scale=3)

data_merge["展销时间"] = data_merge["展销时间"].values.astype("float64")
data_merge["注册日期"] = data_merge["注册日期"].values.astype("float64")
data_merge["上牌日期"] = data_merge["上牌日期"].values.astype("float64")

data_1 = data_merge[data_merge["成交速度"] == 1 ]
data_2 = data_merge[data_merge["成交速度"] == 2 ]
data_3 = data_merge[data_merge["成交速度"] == 3 ]
data_4 = data_merge[data_merge["成交速度"] == 4 ]
data_5 = data_merge[data_merge["成交速度"] == 5 ]
data_6 = data_merge[data_merge["成交速度"] == 6 ]
```

```
from scipy import stats

ab_test = {}
for column in data_merge.columns:
    if data_merge[column].dtype == 'int64' or data_merge[column].dtype == 'float64':
        f,p = stats.f_oneway(data_1[column],data_2[column],data_3[column],data_4[column])
        ab_test[column] = p
#         print(column+":",stats.f_oneway(data_1[column]
#                                         ,data_2[column]
#                                         ,data_3[column]
#                                         ,data_4[column]
#                                         ,data_5[column]
#                                         ,data_6[column]
#                                        ))

a = pd.Series(ab_test)
a = a[a>=0.05]

print(len(a.index))
print(a.index)

for data in [data_1,data_2,data_3,data_4,data_5,data_6]:
    data.drop(columns=a.index,inplace=True)

# 组合训练数据
data = pd.concat([data_1,data_2,data_3,data_4,data_5])

data=data_1

from sklearn import preprocessing # guiyihua
from sklearn.model_selection import train_test_split#划分训练集
from sklearn import metrics
import scipy
from sklearn.feature_selection import RFE
```

```
from sklearn.preprocessing import OrdinalEncoder

encoder = OrdinalEncoder()
# data = data_merge
# data["展销时间"] = data["展销时间"].astype("object")
# data["注册日期"] = data["注册日期"].astype("object")
# data["上牌日期"] = data["上牌日期"].astype("object")

X = data.iloc[:,:-2]
X.drop(columns=['交易价格','最终调价', '车辆级别', '初始降价比', '最终降价比','上架价格', '价格调整'],inplace=True)

# X = encoder.fit_transform(X)

# # X = preprocessing.MinMaxScaler().fit_transform(X)
# # y = data.iloc[:,-2]

# model_1 = XGBRegressor()
# model = model_1
# selector = RFE(model).fit(X, y)
# selector.support_.sum()
# selector.ranking_
# X_wrapper = selector.transform(X)

# column3 = pd.DataFrame([*zip(data.columns,selector.ranking_)]).sort_values(by=1)[:-5].iloc[:,0]
# data_choosen = data[column3.tolist()]

# FR = {}
# data_corr = data_choosen.corr('spearman').abs()
# for i in data_corr.index:
#     listj = []
#     for j in data_corr.columns:
#         if i != j:
#             if data_corr.loc[i,j] >= 0.8:
```

```
#                           listj.append(j)
#         FR[i] = listj

# data_choosen.drop(columns=["匿名特征 5","初始降价比","新车价","调价次数","年款","初始降价","交易价格"],inplace=True)

# X = data_choosen
X = encoder.fit_transform(X)
X = preprocessing.MinMaxScaler().fit_transform(X)

y = np.log1p(data["交易价格"])

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.9,random_state=100)

X.shape

# X = encoder.fit_transform(X)

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=100)


#  模型初始化

model_1 = XGBRegressor()
model_2 = XGBRFRegressor()
model_3 = RandomForestRegressor()

print("***XGBoost***")
model_1 = model_1.fit(X_train,y_train)
y_pred = model_1.predict(X_test)
print("训练集 R2:",model_1.score(X_train,y_train))
print("测试集 R2:",metrics.r2_score(y_test,y_pred))
print("测试集 MSE:",metrics.mean_absolute_error(y_test,y_pred))

model_1 = model_1.fit(X,y)
```

```
X = data_6.iloc[:,:-2]
X.drop(columns=['交易价格','最终调价', '车辆级别', '初始降价比', '最终降价比','上架价格', '价格调整'],inplace=True)
X = encoder.fit_transform(X)
y = np.log1p(data_6["交易价格"])
y_pred = model_1.predict(X)

plt.figure()
plt.plot(range(len(y_pred[1:130])),y_pred[1:130],'b',label="price_predict")
plt.plot(range(len(y_pred[1:130])),y[1:130],'r',label="price_test")
plt.legend(loc="upper right")
plt.show()
```