队伍编号	MCB2103075
赛道	A

二手车估价问题探讨

摘要

随着我国的机动车数量不断增长，人均保有量业随之增加，机动车以“二手车”的形式在流通环节，包括二手车收车、二手车拍卖、二手车零售、二手车置换等环节的流通需求越累越大。本文基于特征处理对影响二手车销量的数据进行分析，并通过皮尔逊系数法计算各种特征对二手车销售周期的影响因素的关键影响。

对于问题一，首先对数据进行处理，对数据中存在的缺失值、异常值进行处理，然后进行特征构造并处理匿名特征，对部分数值型数据进行分桶，最后采用皮尔逊相关系数对特征进行选择处理，之后进行数据标准化、并训练模型，采用集成学习预测二手车的零售交易价格。同时根据问题一中提供的模型评测标准进行构建评估函数，并进行预测与评估结果。

对于问题二，主要任务是对车辆成交周期的分析，需要挖掘影响车辆车轿周期的关键因素，并进一步说明采取什么有效手段，可以加快门店在库车辆的销售速度。针对此问题，本次研究首先对附件四所给数据进行分析与研究，之后结合附件一车辆相关信息对车辆的销售周期进行进一步的分析，在之后，考虑到问题的针对性，采用皮尔逊相关系数对销售周期的影响因素进行更进一步的分析。

对于问题三，我们进行了文献的搜集整理，并根据给出的样本数据，进行分析，分析二手车的销量，价格以及销售周期等因素去提升二手车的流通销售分析，便于二手车的零售市场。

关键词：特征处理，皮尔逊系数法，集成学习

目录

一、问题重述

1.1 问题的背景

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，包括二手车收车、二手车拍卖、二手车零售、二手车置换等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，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[1]带来作用。目前国家并没有出台一个评判二手车资产价值的标准。一些二手车交易平台和二手车第三方估价平台都从自身的角度建立了一系列估价方法用于评估二手车资产的价值。

在一个典型的二手车零售场景，二手车一般通过互联网等线上渠道获取用户线索，线下实体门店对外展销和售卖，俗称 O2O 门店模式。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，二手车商品和其他商品一样，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[2]。

面对二手车市场的供求要求[3]，对二手车数据进行深入的分析，分析相关数据判断二手车的销量，并进行数据分析与建模分析车辆交易周期的影响因素等，解决二手车交易平台在交易中遇到的问题。

1.2 问题的提出

基于针对初赛问题所做的分析与处理，再根据附件所给的数据，通过建立特征处理与数学建模帮助二手车交易平台进行更好的交易：

问题一：基于问题相关数据，构建预测二手车交易价格的训练模型和测试模型，并对附件 2 中的“估价验证数据”进行预测。

问题二：对车辆的成交周期进行分析，挖掘车辆成交周期的关键因素。并分析如果需要加快门店在库车辆的销售速度，可以结合这些关键因素采取哪些手段，并进一步说明这些手段的使用条件和预期效果。

问题三：分析二手车交易过程中，有哪些问题值得研究，可以帮助二手车交易平台进行更好的交易。

二、问题分析

2.1 问题一的分析

问题一是根据以往的数据预测二手车的价格，属于有监督学习里的回归问题，首先查看 36 列变量信息字段的描述，对于时间类型的数据读入数据时转换成时间类型格式，然后查看数据的缺失情况，对缺失严重的特征如 anonymousFeature7 和 anonymousFeature15 的缺失值以达到 50%以上，对这类特征进行删除，并对缺失值不大的特征进行填充如'carCode', 'modelyear', 'gearbox'，由于它们都是分类特征采用众数进行填充，对时间基础周期特征(年月日特征拆解)，同时构建时间差特征如：'tradeTime'-'registerDate' (汽车的使用时间)，对定类数据进行 Frequency 编码，对部分数值型变量进行数据分桶，根据部分特征明显的匿名变量构造新特征，比如 anonymousFeature12（4220*1740*1625）很有可能就是汽车的体积，因此可以构造长宽高三个新的特征，接下来就可以进行特征的选择了，可以使用基于皮尔逊相关系数进行特征选择，然后进行降维可以选择 PCA 主成分分析，然后选择集成学习随机森林训练测试集，最后可以使用 XGboost 优化一下模型。

2.2 问题二的分析

问题二需要结合附件 4“门店交易训练数据”对车辆的成交周期进行分析，挖掘影响车辆成交周期的关键因素。关键因素即一定程度上起决定性作用的影响因素，本报告欲先根据附件 4 对数据进行初步分析，即初步判断哪些因素可能成为关键因素，附件 4 中的价格因素很明显将会成为影响成交周期的关键因素之一，故而在本报告中对于问题二的初步分析的重点就是价格因素，在价格数据记录上做文章进行分析。出于更加全面地考虑，价格差、成交时间、价格变更次数等也将成为初步分析的重点内容。鉴于附件 4 的数据特征过少，不足以说明问题，故将附件 4 同附件 1 进行信息整合，得到较为全面的特征集合，通过进一步挖掘，筛选得出关键因素。为了使得本报告中对于成交周期的影响因素挖掘具有充分说明性，进一步构建模型，即通过特征选择、皮尔逊相关系数计算分析、构建模型，根据现有数据特征对成交周期进行预测，在预测的过程中，不断变更输入的特征子集，观察得分变化。变化较大，即影响较大，从而得出影响成交周期[4]的关键因素。

2.3 问题三的分析

问题三需要依据给出的样本数据集，给出值得研究的问题，即进行问题拓展，并在问题拓展分析之后给出思路。针对现有数据集，查阅、搜集、整理有关二手

车市场[5]的资料，关注二手车买卖中亟待解决的问题，从而提出问题，并试图在本报告中给出思路，予以参考。在查阅大量资料后，本报告欲就淡旺季、车库存货积压[6][7]等方面提出问题并给出思路。

三、模型假设

1.假设销售价格不会超过规定的阈值
2.假设销售的谈判技巧相同或没有显著差异
3.假设这些销售记录都来自同一个平台或者平台间的评判标准类似
4.假设车身受损和维修情况相同

四、符号说明

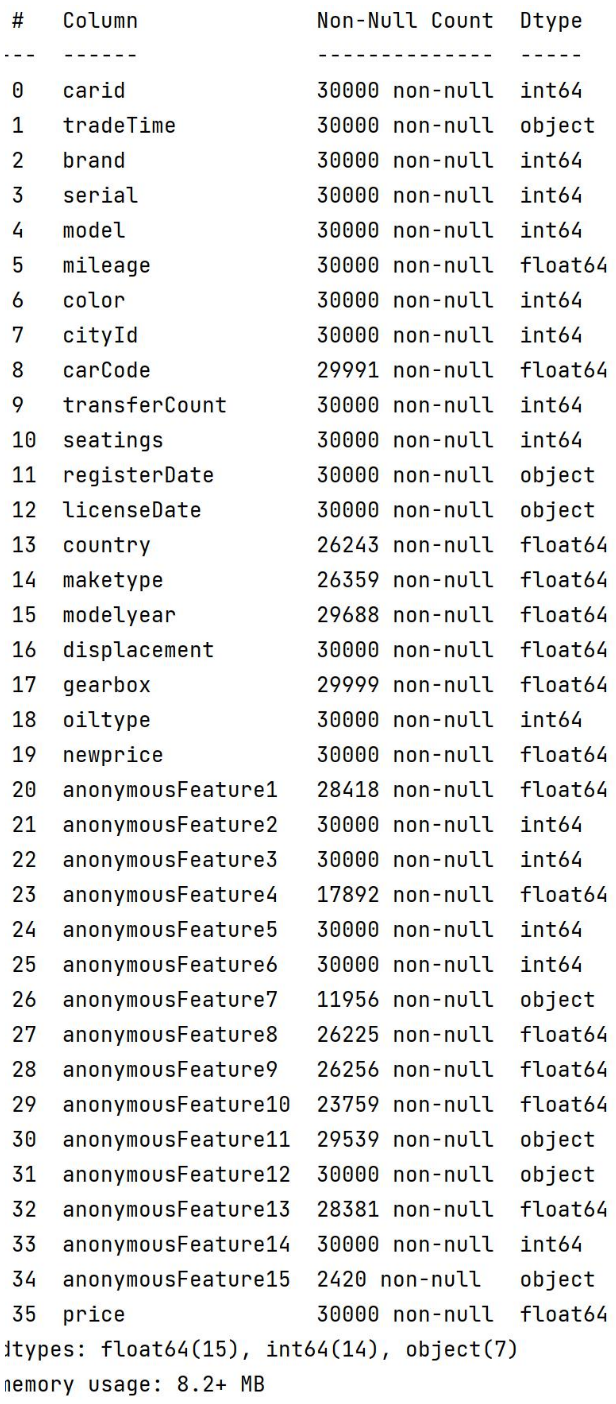
Features	Description
updateTimeX(x∈（1,12）)	第 x 次更新价格时间
updatePriceX(x∈（1,12）)	第 x 次更新对应的价格
updatenum	每条记录对应更新价格次数
withdrawPeriod	计算出的成交周期
withdraw_month	成交月份
dif_price	成交差价
anonymousFeature13_year	匿名特征 13 转化成时间后的年份
anonymousFeature13_month	匿名特征 13 转化成时间后的月份
anonymousFeature12_length	车辆长度
anonymousFeature12_width	车辆宽度
anonymousFeature12_height	车辆高度
old_year	汽车的使用时间
old_year_1	汽车的注册时长
mileage_bin	对mileage进行分箱后的区间段号
decreasing_count	降价的次数

五、基于问题一的研究分析

5.1 数据处理

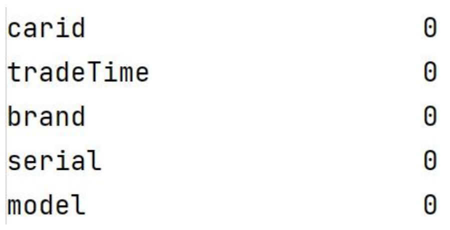
1. 给匿名变量命名，从 anonymousFeature1 到 anonymousFeature15

2. 查看变量数据类型



```
#   Column               Non-Null Count  Dtype
---  ------               --------------  -----
0   carid                30000 non-null  int64
1   tradeTime            30000 non-null  object
2   brand                30000 non-null  int64
3   serial               30000 non-null  int64
4   model                30000 non-null  int64
5   mileage              30000 non-null  float64
6   color                30000 non-null  int64
7   cityId               30000 non-null  int64
8   carCode              29991 non-null  float64
9   transferCount        30000 non-null  int64
10  seatings             30000 non-null  int64
11  registerDate         30000 non-null  object
12  licenseDate          30000 non-null  object
13  country              26243 non-null  float64
14  maketype             26359 non-null  float64
15  modelyear            29688 non-null  float64
16  displacement         30000 non-null  float64
17  gearbox              29999 non-null  float64
18  oiltype              30000 non-null  int64
19  newprice             30000 non-null  float64
20  anonymousFeature1    28418 non-null  float64
21  anonymousFeature2    30000 non-null  int64
22  anonymousFeature3    30000 non-null  int64
23  anonymousFeature4    17892 non-null  float64
24  anonymousFeature5    30000 non-null  int64
25  anonymousFeature6    30000 non-null  int64
26  anonymousFeature7    11956 non-null  object
27  anonymousFeature8    26225 non-null  float64
28  anonymousFeature9    26256 non-null  float64
29  anonymousFeature10   23759 non-null  float64
30  anonymousFeature11   29539 non-null  object
31  anonymousFeature12   30000 non-null  object
32  anonymousFeature13   28381 non-null  float64
33  anonymousFeature14   30000 non-null  int64
34  anonymousFeature15   2420 non-null   object
35  price                30000 non-null  float64
ltypes: float64(15), int64(14), object(7)
nemory usage: 8.2+ MB
```

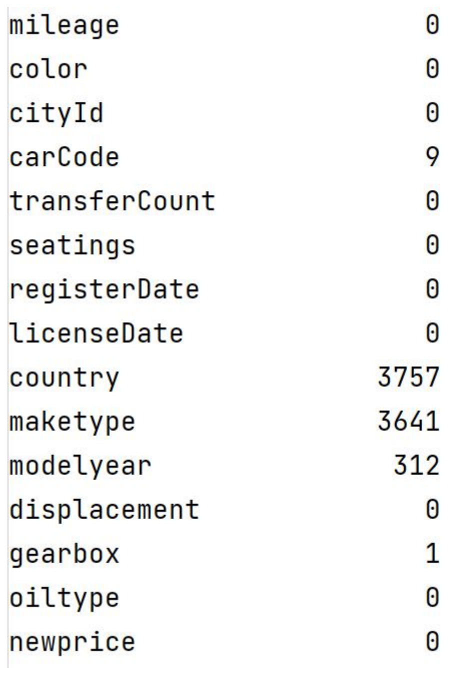
3. 前五列'carid', 'tradeTime', 'brand', 'serial', 'model'都是 id 和时间，不能做中心化操作，而且通过 data.iloc[:, :5].isnull().sum()



```
carid                 0
tradeTime             0
brand                 0
serial                0
model                 0
```

发现没有缺失值，因此不需要处理。

其他列缺失值的情况

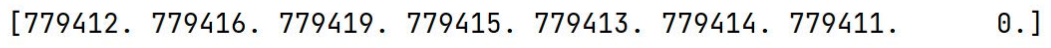


```
mileage                 0
color                   0
cityId                  0
carCode                 9
transferCount           0
seatings                0
registerDate            0
licenseDate             0
country              3757
maketype             3641
modelyear             312
displacement            0
gearbox                 1
oiltype                 0
newprice                0
```

可以看出除了 15 个匿名特征外，carCode 有 9 个缺失值，country 有 3757 个缺失值 maketype 有 3641 个缺失值，modelyear 有 312 个缺失值。gearbox 有一个缺失值。

对于缺失值较少的列，如 carCode 和 gearbox 直接进行删除操作。country ,maketype 和 modelyear 由于缺失值较多，根据数据类型，定类数据选择使用众数填充比较适合。

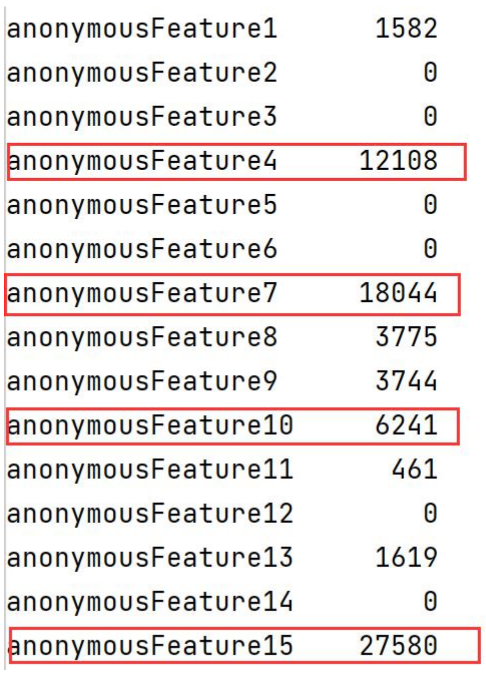
data_1['country'].unique() 可以发现



```
[779412. 779416. 779419. 779415. 779413. 779414. 779411.      0.]
```

这个 0 导致数据量纲较大，会对后期模型的性能产生影响，因此结合所给的信息将其改为 779410

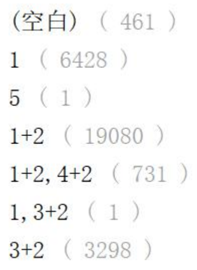
4. 对 15 个匿名变量的分析



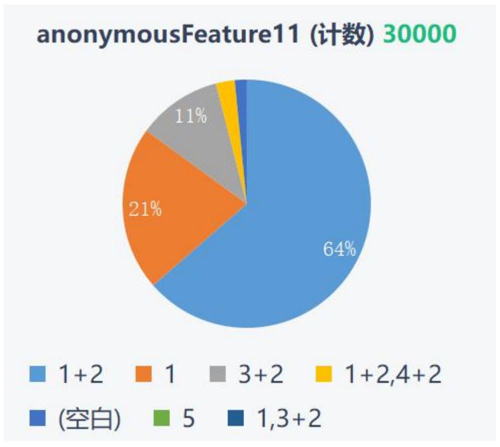
```
anonymousFeature1        1582
anonymousFeature2           0
anonymousFeature3           0
anonymousFeature4       12108
anonymousFeature5           0
anonymousFeature6           0
anonymousFeature7       18044
anonymousFeature8        3775
anonymousFeature9        3744
anonymousFeature10       6241
anonymousFeature11        461
anonymousFeature12          0
anonymousFeature13       1619
anonymousFeature14          0
anonymousFeature15      27580
```

上图圈出的特征由于数据缺失严重直接删除这些特征

①对 anonymousFeature11



```
(空白) ( 461 )
1 ( 6428 )
5 ( 1 )
1+2 ( 19080 )
1+2, 4+2 ( 731 )
1, 3+2 ( 1 )
3+2 ( 3298 )
```



根据显示的信息猜测此特征是某种物品的个数，预测应该为字符串中所有数字之和，使用正则匹配提取所有数值并求和作为新的 anonymousFeature11。

②对 anonymousFeature12

```
2695*1559*1532 ( 3 )
2695*1559*1542 ( 18 )
2695*1559*1555 ( 2 )
2695*1559*1565 ( 46 )
2695*1663*1552 ( 3 )
2695*1663*1555 ( 31 )
3400*1575*1670 ( 6 )
3400*1575*1705 ( 3 )
3450*1575*1675 ( 1 )
3460*1618*1465 ( 32 )
```

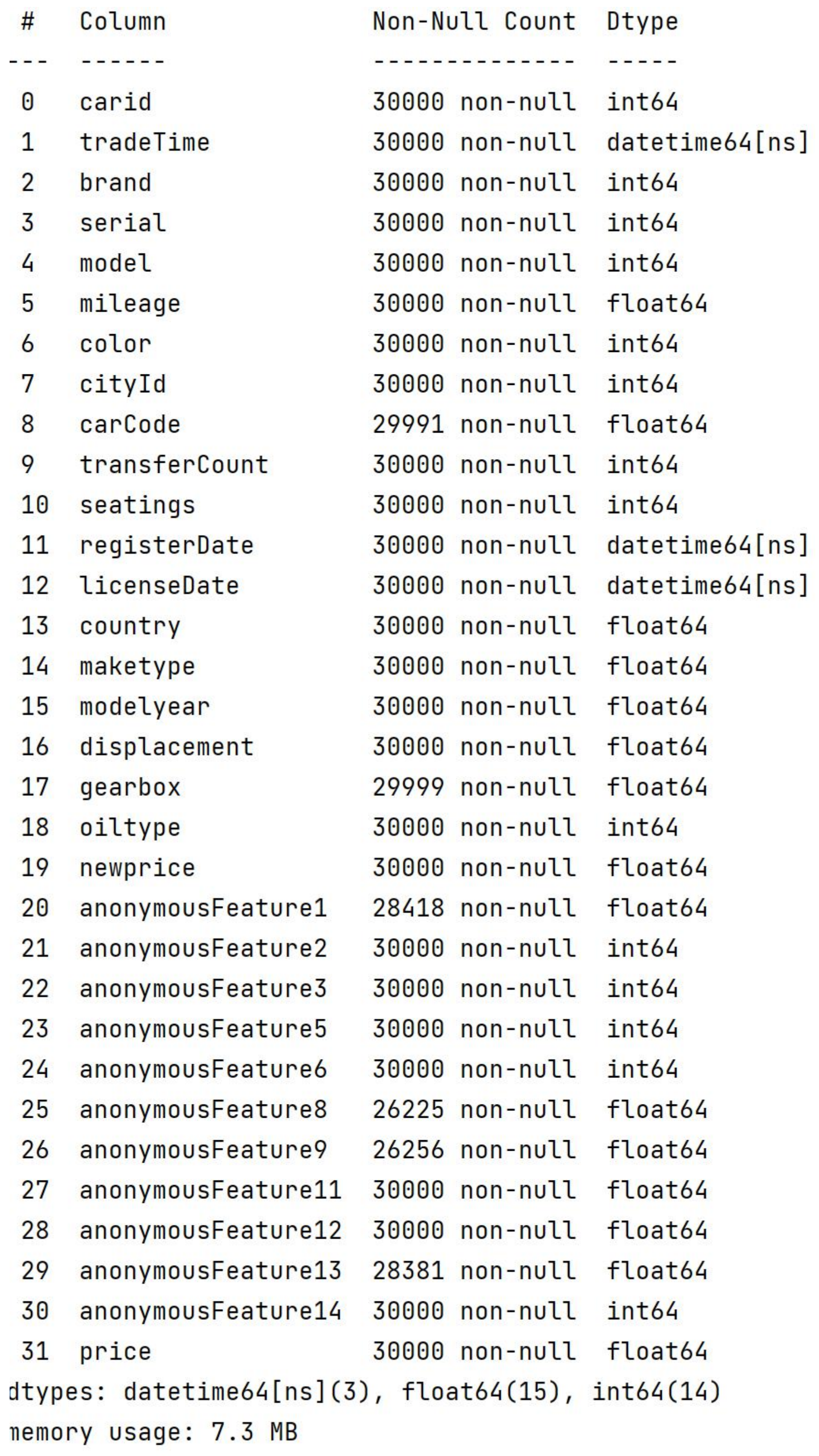
根据其值的表现形式估计其为汽车的体积，所以其值应为三个数的乘积，使用*切分数据并求出乘积做为新的 anonymousFeature12，并用切分得到的三个数据列构建三个新的特征 anonymousFeature12_length ，anonymousFeature12_width，anonymousFeature12_height。

③对 anonymousFeature13

```
200606 ( 18 )
200611 ( 3 )
200612 ( 1 )
200702 ( 6 )
200703 ( 13 )
200705 ( 3 )
200706 ( 2 )
200707 ( 5 )
```

把 anonymousFeature13 转换成字符串进行切片，取前四位为 anonymousFeature13_year，其他为 anonymousFeature13_month。

经过以上处理，数据已经全部变成数值型特征和时间类型的特征



```
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   carid                30000 non-null  int64
 1   tradeTime            30000 non-null  datetime64[ns]
 2   brand                30000 non-null  int64
 3   serial               30000 non-null  int64
 4   model                30000 non-null  int64
 5   mileage              30000 non-null  float64
 6   color                30000 non-null  int64
 7   cityId               30000 non-null  int64
 8   carCode              29991 non-null  float64
 9   transferCount        30000 non-null  int64
 10  seatings             30000 non-null  int64
 11  registerDate         30000 non-null  datetime64[ns]
 12  licenseDate          30000 non-null  datetime64[ns]
 13  country              30000 non-null  float64
 14  maketype             30000 non-null  float64
 15  modelyear            30000 non-null  float64
 16  displacement         30000 non-null  float64
 17  gearbox              29999 non-null  float64
 18  oiltype              30000 non-null  int64
 19  newprice             30000 non-null  float64
 20  anonymousFeature1    28418 non-null  float64
 21  anonymousFeature2    30000 non-null  int64
 22  anonymousFeature3    30000 non-null  int64
 23  anonymousFeature5    30000 non-null  int64
 24  anonymousFeature6    30000 non-null  int64
 25  anonymousFeature8    26225 non-null  float64
 26  anonymousFeature9    26256 non-null  float64
 27  anonymousFeature11   30000 non-null  float64
 28  anonymousFeature12   30000 non-null  float64
 29  anonymousFeature13   28381 non-null  float64
 30  anonymousFeature14   30000 non-null  int64
 31  price                30000 non-null  float64
dtypes: datetime64[ns](3), float64(15), int64(14)
nemory usage: 7.3 MB
```

5. 处理定类特征

对 color，carCode，country，modelyear 这些类别行的特征，因为是定类数据，数据之间本来应该是没有相对大小的，但是转为数值后比如 1，2，3 会有相对大小，会对树模型产生不好的影响。因此本组选择使用 Frequency 编码对定类数据进行转换，Frequency 编码通过计算特征变量中每个值的出现次数来表示

该特征的信息。

6. 再用时间类型的特征构建新特征

①基础周期特征的拆除（年月日特征拆解）

data['tradeTime_year']=data['tradeTime'].dt.year
data['tradeTime_month']=data['tradeTime'].dt.month
data['tradeTime_day']=data['tradeTime'].dt.day
data['registerDate_year']=data['registerDate'].dt.year
data['registerDate_month']=data['registerDate'].dt.month
data['registerDate_day']=data['registerDate'].dt.month

②时间差

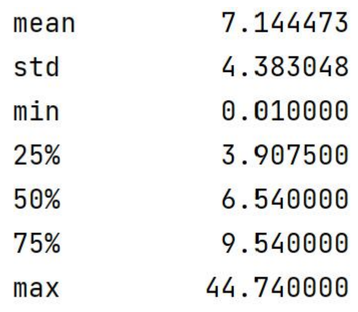
‘old_year’='tradeTime'-'registerDate' (汽车的使用时间)

‘old_year_1’='tradeTime'-'licenseDate'

经过以上处理后，特征以全部转换成数据值

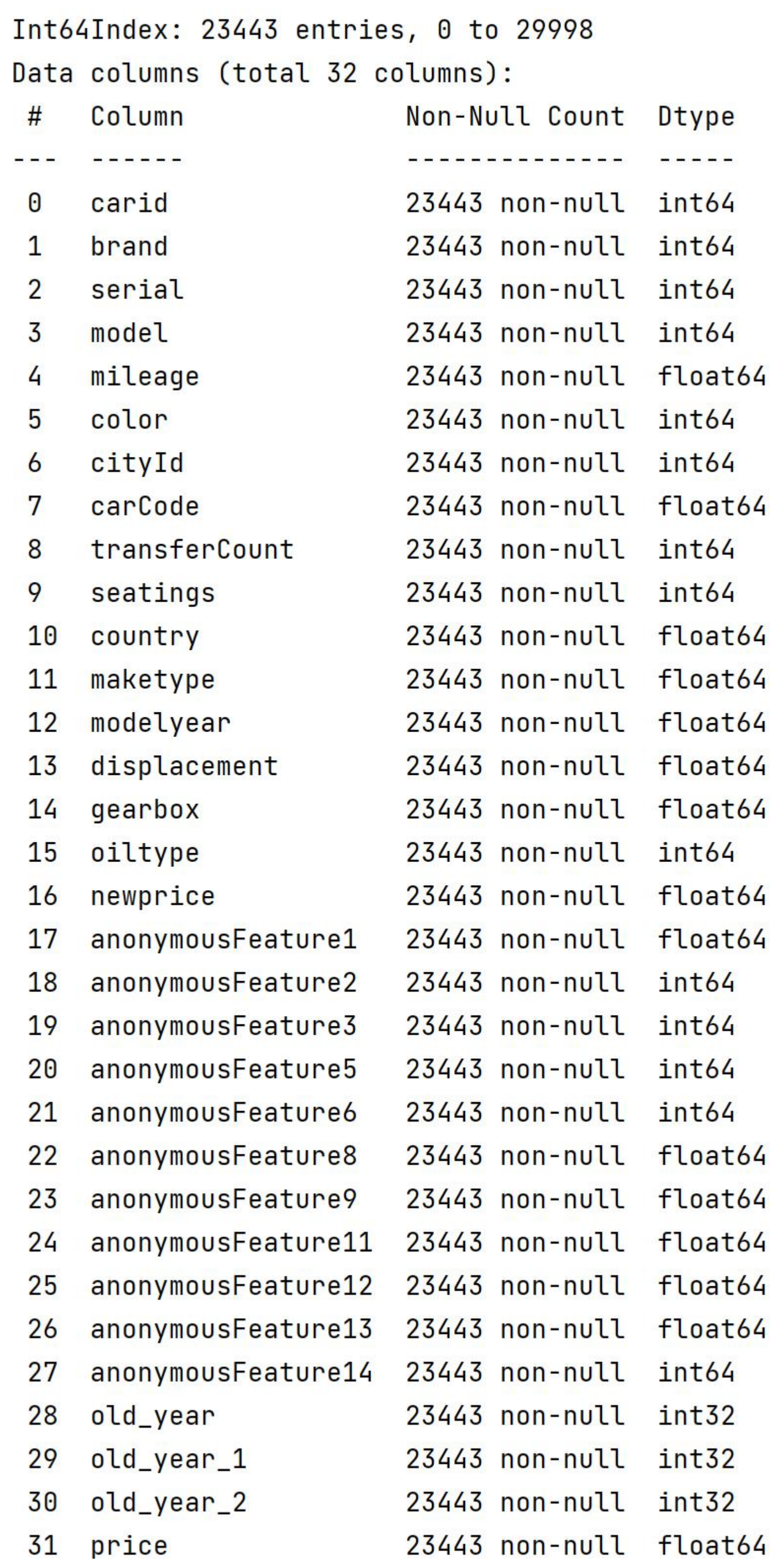
7. 数值特征[8]的处理

数据分桶，对 mileage 特征进行数据分桶,mileage 的均值为 7.14，其余统计量如下



```
mean         7.144473
std          4.383048
min          0.010000
25%          3.907500
50%          6.540000
75%          9.540000
max         44.740000
```

因此分组区间为 (0,1],(1,4],(4,7.15],(7.15,10],(10,50]



```
Int64Index: 23443 entries, 0 to 29998
Data columns (total 32 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   carid               23443 non-null  int64  
 1   brand               23443 non-null  int64  
 2   serial              23443 non-null  int64  
 3   model               23443 non-null  int64  
 4   mileage             23443 non-null  float64
 5   color               23443 non-null  int64  
 6   cityId              23443 non-null  int64  
 7   carCode             23443 non-null  float64
 8   transferCount       23443 non-null  int64  
 9   seatings            23443 non-null  int64  
 10  country             23443 non-null  float64
 11  maketype            23443 non-null  float64
 12  modelyear           23443 non-null  float64
 13  displacement        23443 non-null  float64
 14  gearbox             23443 non-null  float64
 15  oiltype             23443 non-null  int64  
 16  newprice            23443 non-null  float64
 17  anonymousFeature1   23443 non-null  float64
 18  anonymousFeature2   23443 non-null  int64  
 19  anonymousFeature3   23443 non-null  int64  
 20  anonymousFeature5   23443 non-null  int64  
 21  anonymousFeature6   23443 non-null  int64  
 22  anonymousFeature8   23443 non-null  float64
 23  anonymousFeature9   23443 non-null  float64
 24  anonymousFeature11  23443 non-null  float64
 25  anonymousFeature12  23443 non-null  float64
 26  anonymousFeature13  23443 non-null  float64
 27  anonymousFeature14  23443 non-null  int64  
 28  old_year            23443 non-null  int32  
 29  old_year_1          23443 non-null  int32  
 30  old_year_2          23443 non-null  int32  
 31  price               23443 non-null  float64
```

8. 清除数据中的异常值

首先查看数据的分布，找出异常值

```
data_2.describe()
```

	old_year_1	old_year_2	price
count	23443.000000	23443.000000	23443.000000
mean	2013.102419	114.041676	18.249138
std	960.021532	120.010169	711.944332
min	12.000000	-2104.000000	0.050000
25%	1239.000000	49.000000	6.180000
50%	1894.000000	81.000000	9.980000
75%	2722.500000	135.000000	17.380000
max	5102.000000	2559.000000	109000.000000

可以明显的看出，price 有异常值，接下来用箱线图检测异常值，并构造函数处理异常值

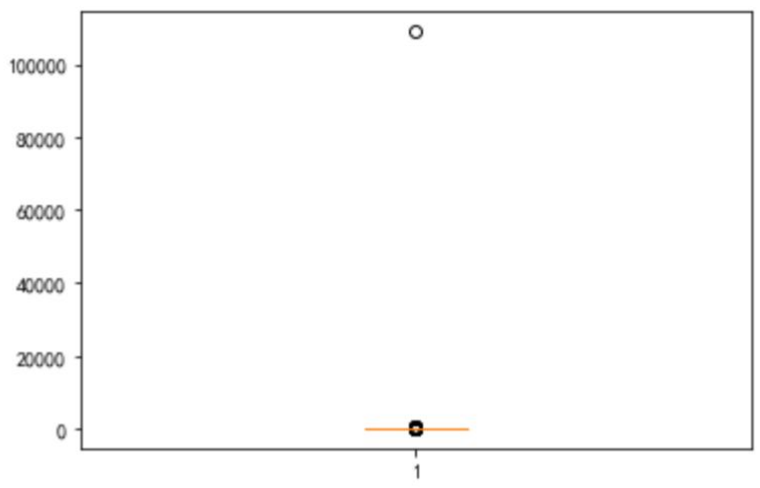


图 1　Price 的箱线图

从图中我们也可以看出，price 明显存在异常值。

5.2 模型架构

1. 从这些特征中选择出最合适的特征（特征选择），为模型构建做准备。

皮尔逊相关系数的特征选择[9]

```
pearson=data_2.corr()
index=pearson['price'][:-1].abs() > 0.005
X=data_2.iloc[:,:-1]
```

```
carid                          0.012330
brand                          0.124546
serial                         0.157681
model                          0.092608
mileage                       -0.122911
color                         -0.054575
cityId                        -0.113928
carCode                       -0.056992
transferCount                  0.076590
seatings                       0.011833
country                       -0.231646
maketype                       0.389555
modelyear                      0.238235
displacement                   0.534374
gearbox                        0.132481
oiltype                        0.012849
newprice                       0.750068
anonymousFeature1              0.006181
anonymousFeature2              0.496486
anonymousFeature3             -0.015633
anonymousFeature5              0.232440
anonymousFeature6              0.060729
anonymousFeature8              0.444966
anonymousFeature9             -0.052654
anonymousFeature11             0.462009
anonymousFeature12             0.393618
anonymousFeature13             0.224887
anonymousFeature14            -0.036081
anonymousFeature13_year        0.224241
anonymousFeature13_month       0.032308
anonymousFeature12_length      0.394336
anonymousFeature12_width       0.485100
anonymousFeature12_height      0.128487
old_year                      -0.217594
old_year_1                    -0.242086
old_year_2                     0.199830
Name: price, dtype: float64
```

2. 进行数据降维（PCA 主成分分析）

主成分分析(PCA,principal components analysis)将高维数据投影至低维空间，低维空间中的新特征叫主成分/超级列[10]。只需要几个主成分就可以准确解释整个数据集。

（1）创建数据集的协方差矩阵;

（2）计算协方差矩阵的特征值;

（3）保留前 n 个特征值&特征向量(按特征值降序排列);

（4）用保留的特征向量转换新的数据点。

3. 数据标准化

数据标准化是而在多指标评价体系中，由于各评价指标的性质不同，通常具

有不同的量纲和数量级。当各指标间的水平相差很大时，如果直接用原始指标值进行分析，就会突出数值较高的指标在综合分析中的作用，相对削弱数值水平较低指标的作用。因此，为了保证结果的可靠性，需要对原始指标数据进行标准化处理。这里采用的标准差标准化

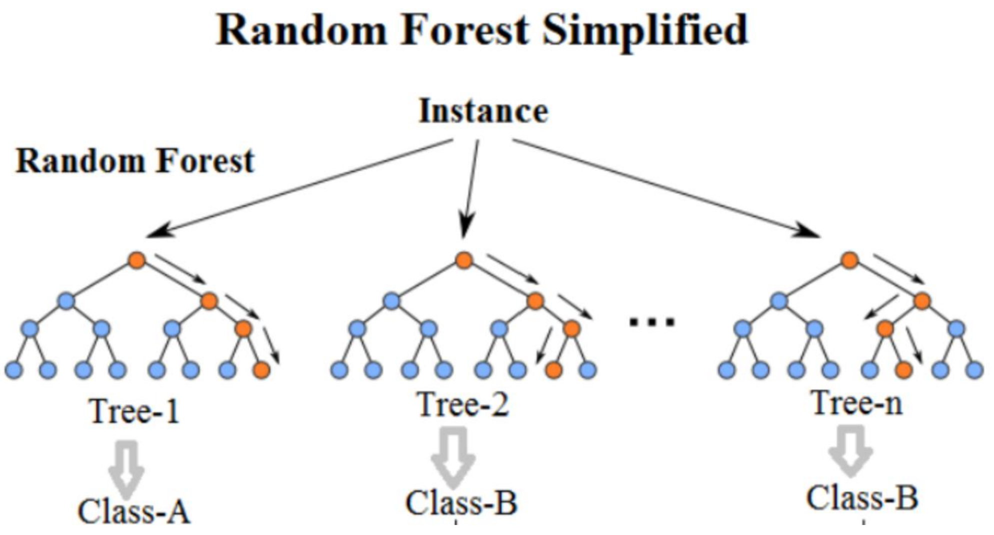
标准差标准化也叫零均值标准化或分数标准化,是当前使用最广泛的数据标准化方法。经过该方法处理的数据均值为 0 ,标准差为 1，转化公式如下。

$$X^* = \frac{X-\bar{X}}{\delta}$$

其中 X 为原始数据的均值, δ为原始数据的标准差。标准差标准化后的值区间不局限于[0,1] ,并且存在负值。同时也不难发现,标准差标准化和离差标准化- -样不会改变数据的分布情况。

4. 训练模型

综合考虑各方面因素后选择集成学习里的随机森林进行模型的训练，随机森林就是使用多个决策树算法，创建弱评估器，其基本原理如下



而决策数进行特征选择和划分的时候主要依据信息增益，这里要引入部分信息论的知识，信息的计算公式如下

$$\mathrm{I}(xi) = -\log_2 p(x_i) = \log_2 \frac{1}{p(x_i)}$$

而熵[11]就是信息的期望，其计算公式如下

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$$

然后还需要计算一下条件熵（条件熵 H（Y|X）表示在已知随机变量 X 的条件下随机变量 Y 的不确定性）

$$H(Y|X)=\sum_{x\in X}p(x)H(Y|X=x)$$

最后需要计算的就是要求的信息增益

信息增益是：信息熵-条件熵。

而决策数在进行特征选择的时候就是根据信息增益如果 IG（信息增益大）的话那么这个特征对于分类来说很关键，决策树就是这样来找特征的。随机森林就是所有决策树平均值[12]。

5. 进行模型的预测

使用训练好的模型将测试集的数据放入进行预测

6. 得出预测结果后利用评估函数进行评价，评判模型好坏

评估函数

$$0.2*(1-Mape)+0.8*Accuracy_5$$

Ape(相对误差)：

$$Ape=\left|\hat{y}-y\right|\Big/y$$

Mape(平均相对误差)：

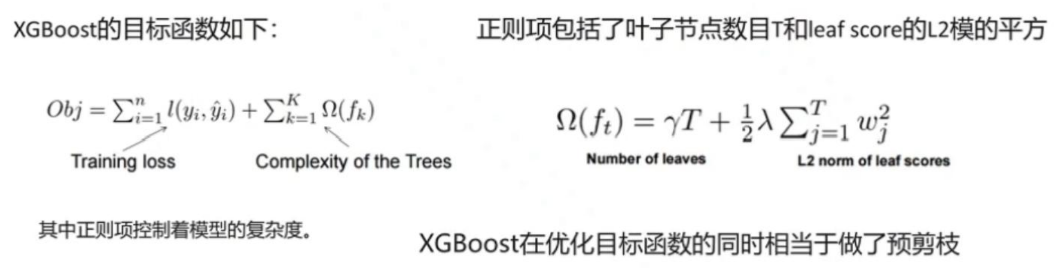
$$Mape=\frac{1}{m}\sum_{i=1}^{m}Ape_i$$

其中，真实值$y=(y_1,y_2,..,y_m)$，模型预测值为$y=(y_1,y_2,..,y_m)$；

$Accuracy_5$(5%误差准确率):

$$Accuracy_5=count(Ape<=0.05)/count(total)$$

7. 最后使用 XGboost 优化一下模型


XGBoost的目标函数如下：

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training loss　Complexity of the Trees

其中正则项控制着模型的复杂度。

正则项包括了叶子节点数目T和leaf score的L2模的平方

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Number of leaves　L2 norm of leaf scores

XGBoost在优化目标函数的同时相当于做了预剪枝

XGBoost 的优势:
1. 正则化
可有效防止模型过拟合
2. 并行处理
XGBoost 并行处理数据的速度要比普通提升树算法更快.
3. 自动处理缺失数据
不做切分点，但会把缺失值放进左右子树看效果
4. 剪枝策略
普通的提升采用的是贪心算法，只有在不再有增益时才会停止分裂
5. 可以在目前树模型上继续迭代
XGBoost 可以在训练中使用不同的迭代策略[13]。

六、基于问题二的研究分析

6.1 影响因素初步探索

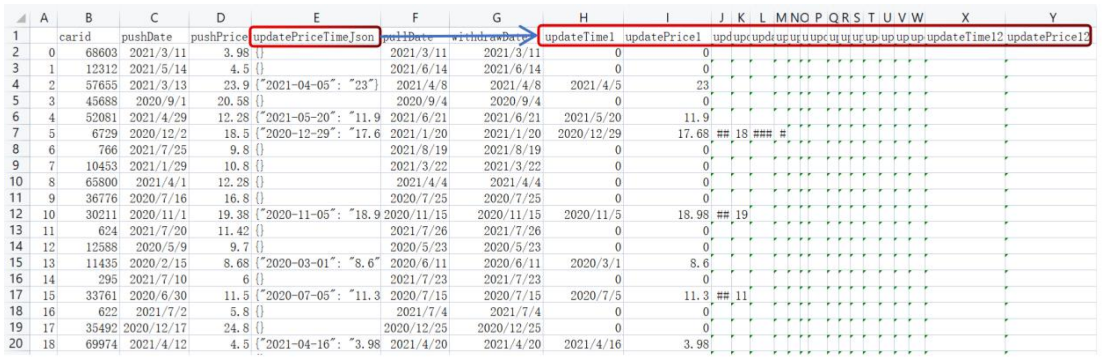
在此过程中，考虑到文件对附件四的要求，首要对附件四相关数据进行处理分析，并对影响因素进行分析处理。

6.1.1 数据处理

1. **说明：**对影响因素的初步探索主要是针对附件 4 进行数据探索与分析。
2. **处理思路与方法：**

（1）提取每次变更价格的时间和此次对应价格

经过统计，变更价格存在 0、1、2、3、4、5、6、7、8、12 的频次数量，故首先对每一次的更新做信息提取，由“updatePriceTimeJson”属性列提取出每一次的更新时间和对应价格，即由“updatePriceTimeJson”属性列提取出“updateTime1”、“updatePrice1”.....“updateTime12”、“updatePrice12”。截取提取出的部分信息如下图所示：



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1		carid	pushDate	pushPrice	updatePriceTimeJson	pullDate	withdrawDate	updateTime1	updatePrice1	upd	upc	upda	up	u	u	upc	u	u	u	up	up	up	up	updateTime12	updatePrice12
2	0	68603	2021/3/11	3.98	{}	2021/3/11	2021/3/11	0	0																
3	1	12312	2021/5/14	4.5	{}	2021/6/14	2021/6/14	0	0																
4	2	57655	2021/3/13	23.9	{"2021-04-05": "23"}	2021/4/8	2021/4/8	2021/4/5	23																
5	3	45688	2020/9/1	20.58	{}	2020/9/4	2020/9/4	0	0																
6	4	52081	2021/4/29	12.28	{"2021-05-20": "11.9	2021/6/21	2021/6/21	2021/5/20	11.9																
7	5	6729	2020/12/2	18.5	{"2020-12-29": "17.6	2021/1/20	2021/1/20	2020/12/29	17.68	##	18	###	#												
8	6	766	2021/7/25	9.8	{}	2021/8/19	2021/8/19	0	0																
9	7	10453	2021/1/29	10.8	{}	2021/3/22	2021/3/22	0	0																
10	8	65800	2021/4/1	12.28	{}	2021/4/4	2021/4/4	0	0																
11	9	36776	2020/7/16	16.8	{}	2020/7/25	2020/7/25	0	0																
12	10	30211	2020/11/1	19.38	{"2020-11-05": "18.9	2020/11/15	2020/11/15	2020/11/5	18.98	##	19														
13	11	624	2021/7/20	11.42	{}	2021/7/26	2021/7/26	0	0																
14	12	12588	2020/5/9	9.7	{}	2020/5/23	2020/5/23	0	0																
15	13	11435	2020/2/15	8.68	{"2020-03-01": "8.6"	2020/6/11	2020/6/11	2020/3/1	8.6																
16	14	295	2021/7/10	6	{}	2021/7/23	2021/7/23	0	0																
17	15	33761	2020/6/30	11.5	{"2020-07-05": "11.3	2020/7/15	2020/7/15	2020/7/5	11.3	##	11														
18	16	622	2021/7/2	5.8	{}	2021/7/4	2021/7/4	0	0																
19	17	35492	2020/12/17	24.8	{}	2020/12/25	2020/12/25	0	0																
20	18	69974	2021/4/12	4.5	{"2021-04-16": "3.98	2021/4/20	2021/4/20	2021/4/16	3.98																

（2）获取每一次记录对应的变更价格次数

仍然由“updataPriceTimeJson”属性列统计得到附件四中每一条记录所对应车辆的价格变更次数（更新次数）。更新次数统计如下图：

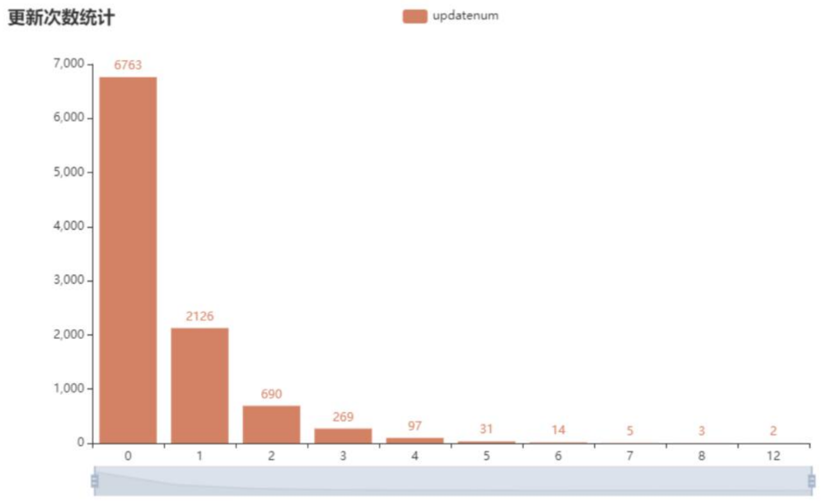
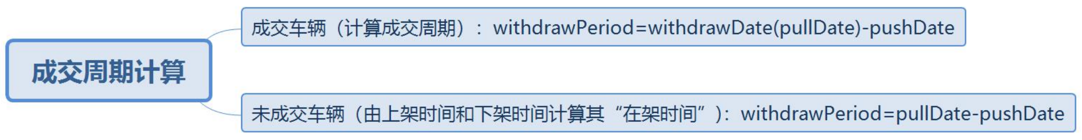


图 2 车辆价格更新次数统计图

由图可以看出，绝大多数二手车并不存在对价格的调整。相对来讲，调整价格的二手车占据少数（调整价格的原因在报告下文有分析，此处不赘述），即使调整价格，也大多为 1、2、3 次，调整价格超过三次的车辆占据极少数。

（3）计算得到成交周期

针对成交车辆和未成交车辆分别进行计算：


成交周期计算
成交车辆（计算成交周期）：withdrawPeriod=withdrawDate(pullDate)-pushDate
未成交车辆（由上架时间和下架时间计算其“在架时间”）：withdrawPeriod=pullDate-pushDate

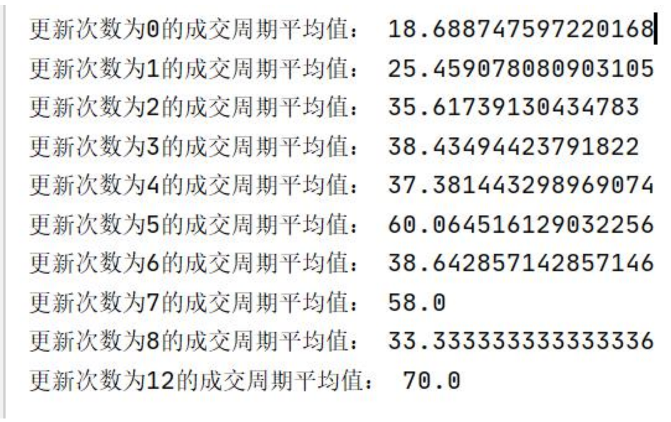
（4）计算成交差价（成交差价 = 成交价格 - 上架价格）

考虑到成交差价对成交周期的影响，由成交价格（即调整到最新的价格）减去该车辆上架时的价格得到成交差价，便于下一步的分析。

6.1.2 价格变更次数对成交周期的影响分析

1. 价格变更次数之于成交周期的影响

考虑到买家在购入二手车时，如果不急于用车，会持观望态度，即等待降价。而卖家可能因已上架车辆迟迟未卖出进行降价或者因市场需求原因对已上架车辆进行升价。由此探索价格变动频次对成交周期的影响。结果如下：



```
更新次数为0的成交周期平均值： 18.688747597220168
更新次数为1的成交周期平均值： 25.459078080903105
更新次数为2的成交周期平均值： 35.61739130434783
更新次数为3的成交周期平均值： 38.43494423791822
更新次数为4的成交周期平均值： 37.381443298969074
更新次数为5的成交周期平均值： 60.064516129032256
更新次数为6的成交周期平均值： 38.642857142857146
更新次数为7的成交周期平均值： 58.0
更新次数为8的成交周期平均值： 33.333333333333336
更新次数为12的成交周期平均值： 70.0
```

由计算结果可以看到，价格更新次数之于成交周期的大致趋势：价格更新频次越频繁，成交周期越长。若当前市场该车需求不大且消费者不急于购入二手车，就会持有观望态度，等待车辆降价，以达到最省钱的目的。站在卖家的角度，如果车辆长期未卖出，会导致车库中的车辆积压的情况，故被迫削减利益，更新降价。

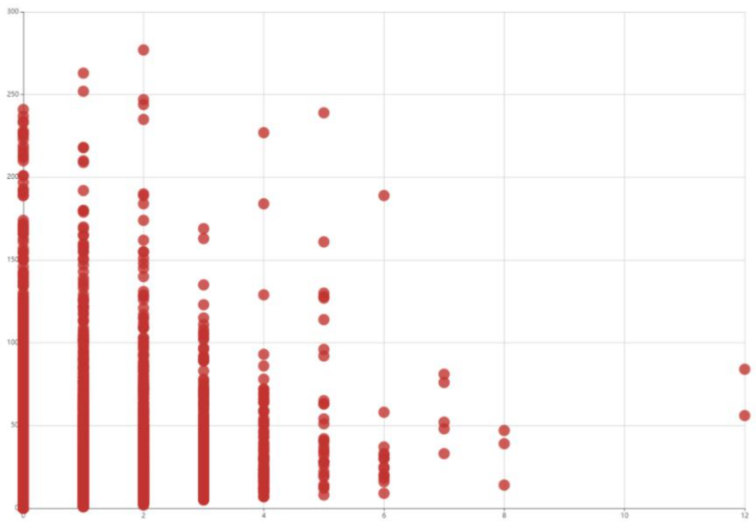


图 3 价格更新趋势

2. 价格变更次数之于车辆成交与否的影响

频繁变更价格对于车辆能否成交存在着一定的关系，统计结果如下：

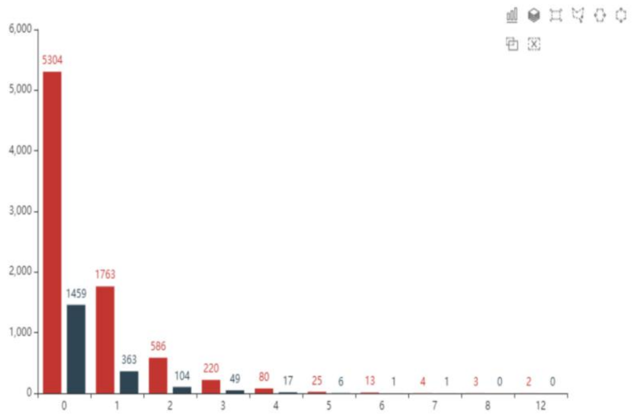
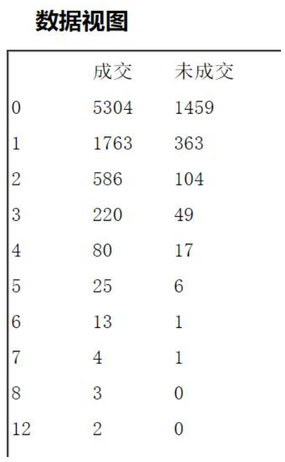
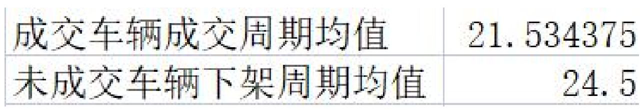


图 4 价格变更与车辆成交



数据视图

	成交	未成交
0	5304	1459
1	1763	363
2	586	104
3	220	49
4	80	17
5	25	6
6	13	1
7	4	1
8	3	0
12	2	0



成交车辆成交周期均值	21.534375
未成交车辆下架周期均值	24.5

由统计结果可以看出，无论是成交车辆还是未成交车辆，未变更价格的车辆都占据大多数。由数据视图中的数据可以看到，坐实了上文分析的情况，即当车辆长期积压时，卖家通过频繁变更价格降价，此时买家就会看准时机，出价购车。又根据成交车辆和未成交车辆的成交周期均值，相差无几。

6.1.3 上架价格对成交周期的影响分析

1.二手车价格分布情况

首先通过散点图得到二手车价格的分布情况，由图可以看到，符合对二手车价格的预期，相对来讲价格较低的占据多数。

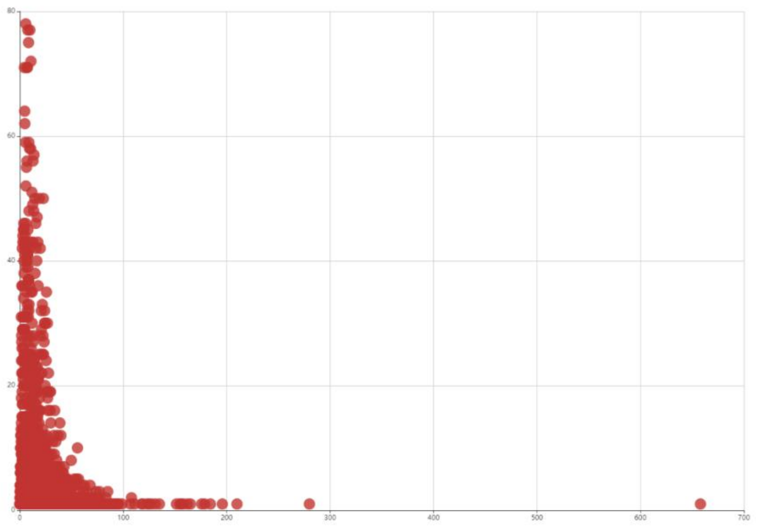
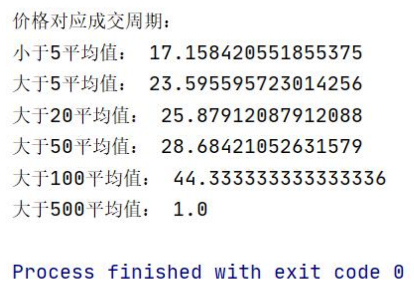


图 5 二手车价格分布情况

2.上架价格对成交周期的影响分析



```
价格对应成交周期:
小于5平均值:  17.158420551855375
大于5平均值:  23.595595723014256
大于20平均值:  25.87912087912088
大于50平均值:  28.68421052631579
大于100平均值:  44.333333333333336
大于500平均值:  1.0

Process finished with exit code 0
```

由统计结果，可以看到车辆价格越低，成交周期越短。结合之前分析的价格变更次数可知，车辆价格越低越容易卖出去，亦即成交周期相对较短。

6.2 结合附件一的数据分析

在对任务二车辆的交易周期的分析过程中，本次研究考虑到实际销售过程中存在的因为车的自身特征，如品牌、车系、动力以及车况等因素对二手车的销售周期[14]的影响，综合附件一与附件四的数据信息对影响车辆交易周期的关键因素进行分析挖掘。

6.2.1 数据分析

在本次研究中，主要研究对象为商品的销售周期，而在文件中对销售周期的计算为出售日期-上架日期。其次考虑到日期的变化性，我们对车辆的销售周期进行分组分析，以便进一步的实现数据的观察与处理。对销售周期进行分组时，我们分为销售周期天数为 0， 1， 2， 3， 4， 5-7， 8-17， 18-50， 51-91，

92-237，这样可以帮助我们更好的整理数据之间的关系，并进行数据分析。

6.2.2 分析车辆相关信息对销售周期天数的影响

在本次研究中，通过对车辆的相关信息进行销售周期天数的研究分析[15]，对于车辆的本身信息，主要从车辆品牌、颜色、燃油类型等方面进行了分析

首先，在进行车辆相关信息对销售周期天数的影响时，我们首先分析未修改的价格的车辆信息，通过绘制不同的销售周期（天）的散点图进行了数据的分析于观察，从图中可以看出明显销售周期短的车辆的销售情况好，对于出售的二手车而言，在较短周期天数内出售的车辆较多。

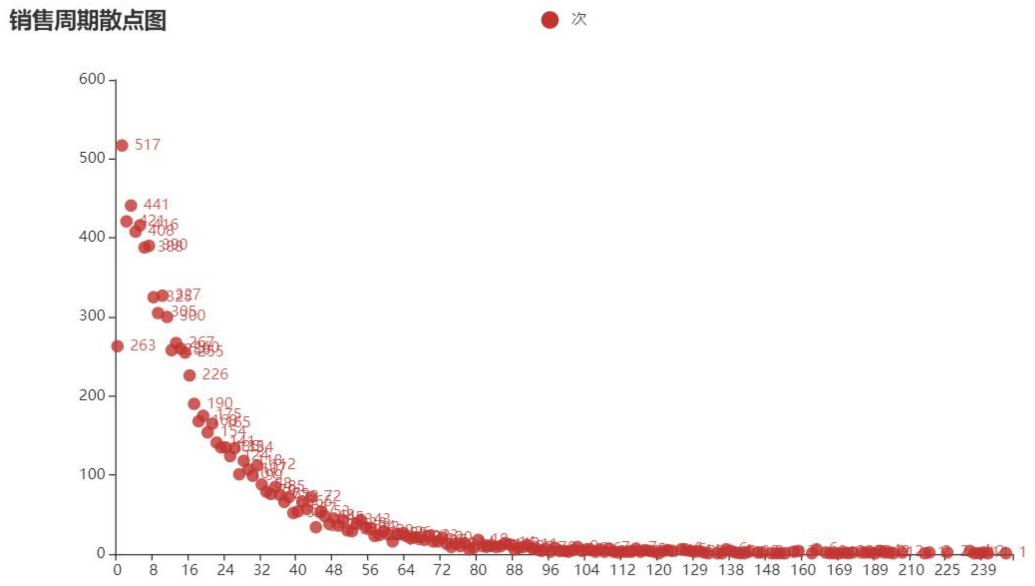


图 6 车辆的销售周期

其次，我们分析了品牌 id 对销售周期的影响，通过观察每个品牌的销售量以及每个天数的销售周期，并对前十数据进行绘图分析，通过每个品牌的销售量分析每个品牌的受欢迎程度；由于品牌量过多，无法一一比较个品牌的销售周期，暂取销售量最多和销售量最少的十个品牌进行销售周期天数的分析于整理，进行观察品牌对销售量的影响。

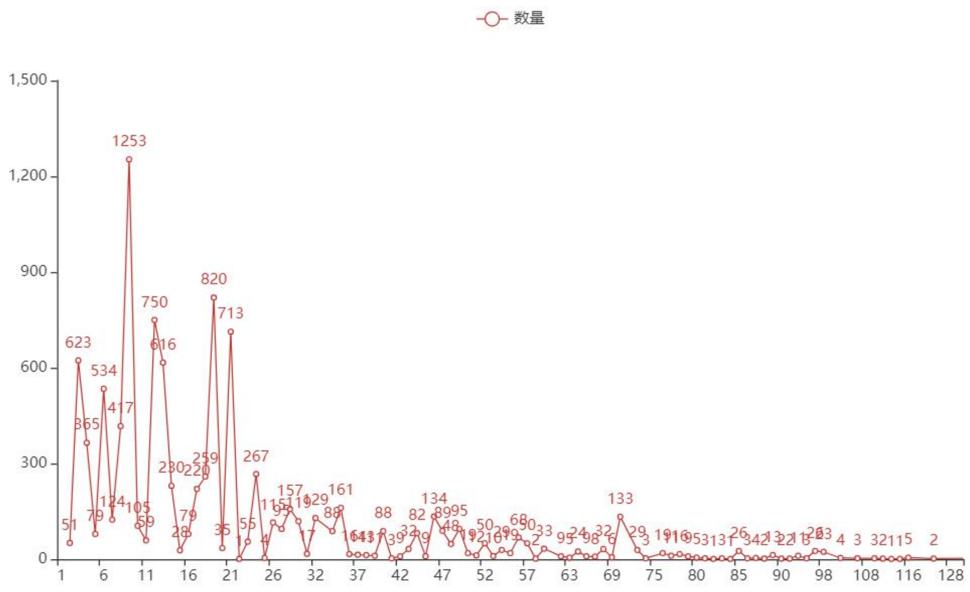


图 7 各品牌的二手车销售量

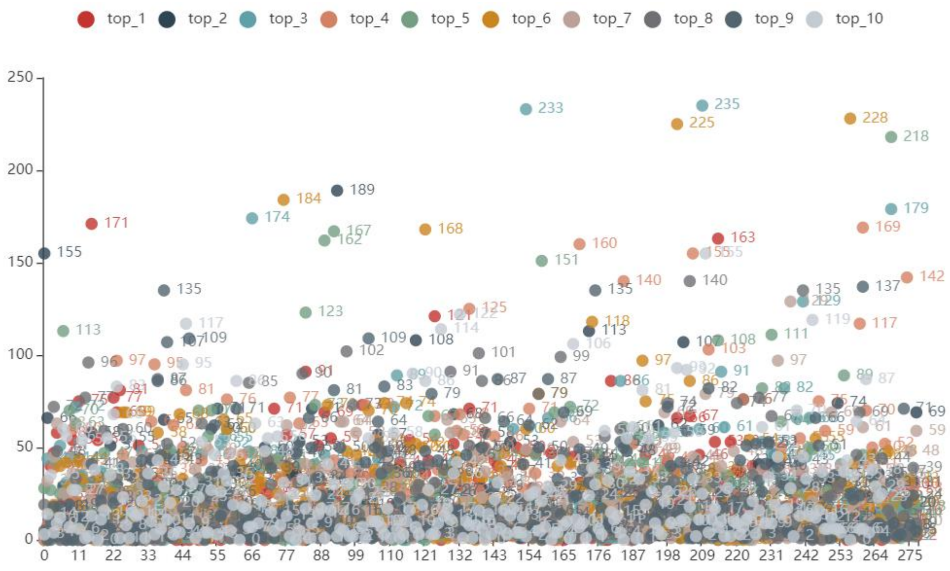


图 8 品牌销售量前十的销售天数分析

从图中我们观察到品牌对销售量的影响居于显著作用，好的品牌可以助推二手车的销售量，然而对于不同车辆的销售周期而言，好的品牌夜有销售周期天数较长的存在，故而品牌对二手车的销售周期天数无显著影响。

之后，在对车系、车型 id、里程、车辆所在城市 id、排量等种类较多的情况进行了如品牌的取销量前十与后十的方式进行了分析。实验过程中，对该类数据的分析如下图所示：

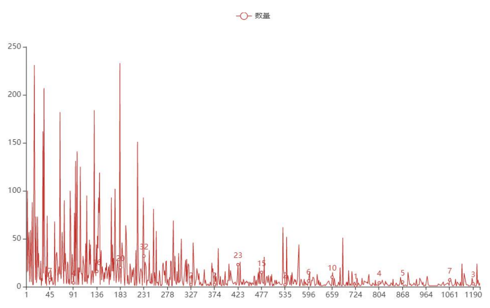


图 9 车系销量分布情况

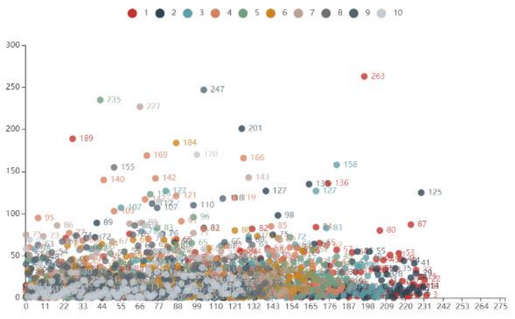


图 10 车系销量前十的销售周期天数情况

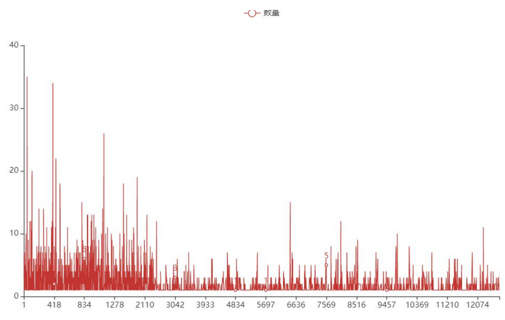


图 11 车型 id 的销售周期天数基本情况

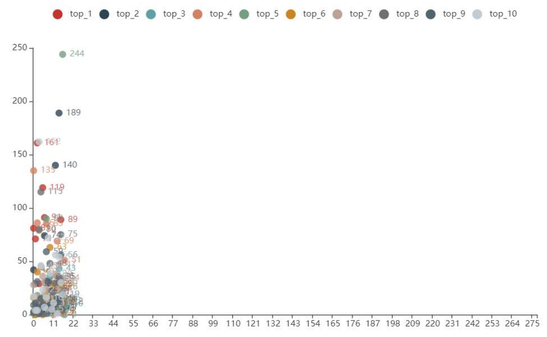


图 12 里程的前十的销售周期前十的基本情况

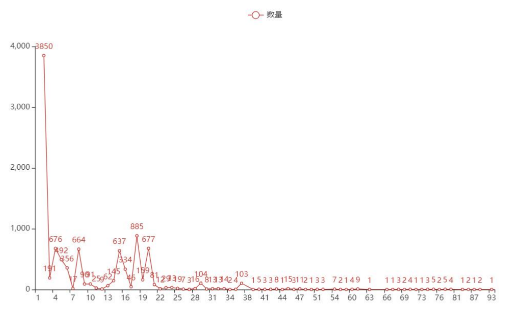


图 13 车辆所在城市 id 销量分布图

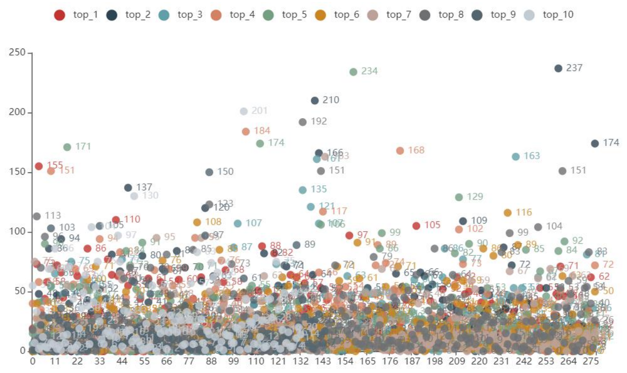
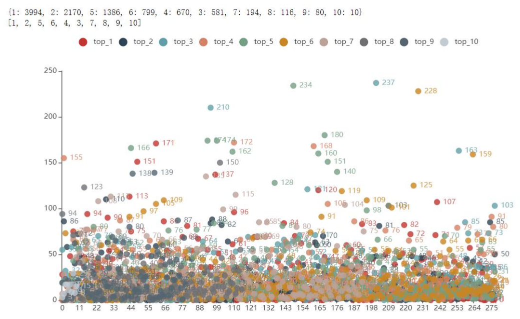


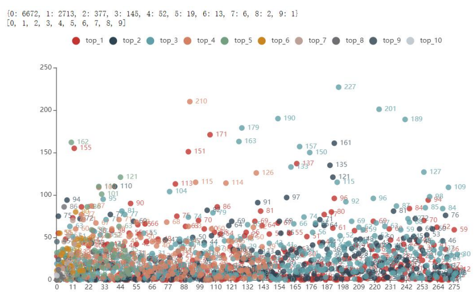
图 14 排量销量前十对销售周期天数的影响分布图

对于车辆颜色、过户次数、燃油类型这种种类较少的则采用整体分析，观察其不同的种类对销售周期天数的影响



```
{1: 3994, 2: 2170, 5: 1386, 6: 799, 4: 670, 3: 581, 7: 194, 8: 116, 9: 80, 10: 10}
[1, 2, 5, 6, 4, 3, 7, 8, 9, 10]
```

图 15 车辆颜色对销售周期天数的影响



```
{0: 6672, 1: 2713, 2: 377, 3: 145, 4: 52, 5: 19, 6: 13, 7: 6, 8: 2, 9: 1}
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

图 16 过户次数的销售周期天数分布图

```
{1: 9601, 2: 350, 3: 39, 5: 7, 4: 3}
[1, 2, 3, 5, 4]
```

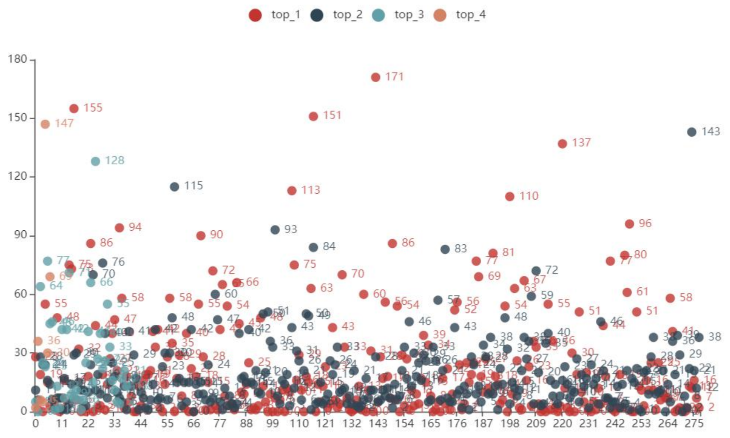


图 17 燃油类型对车辆销售周期天数的影响

6.2.3 对销售周期进行分区间进行统计整理

通过对销售周期进行分区间整理，通过对销售周期进行分组，可以更好的实现对销售周期的分析。对销售周期进行分组时，我们分为销售周期天数为 0， 1， 2， 3， 4， 5-7， 8-17， 18-50， 51-91， 92-237，这样可以帮助我们更好的整理数据之间的关系，并进行数据分析。在进行区间分析的时候，我们参考了 2.2.1 的处理方式，并进行了简略版的处理与分析。

首先，通过对销售周期区间的大致数量分析，让我们更好的了解数据图谱的分析。

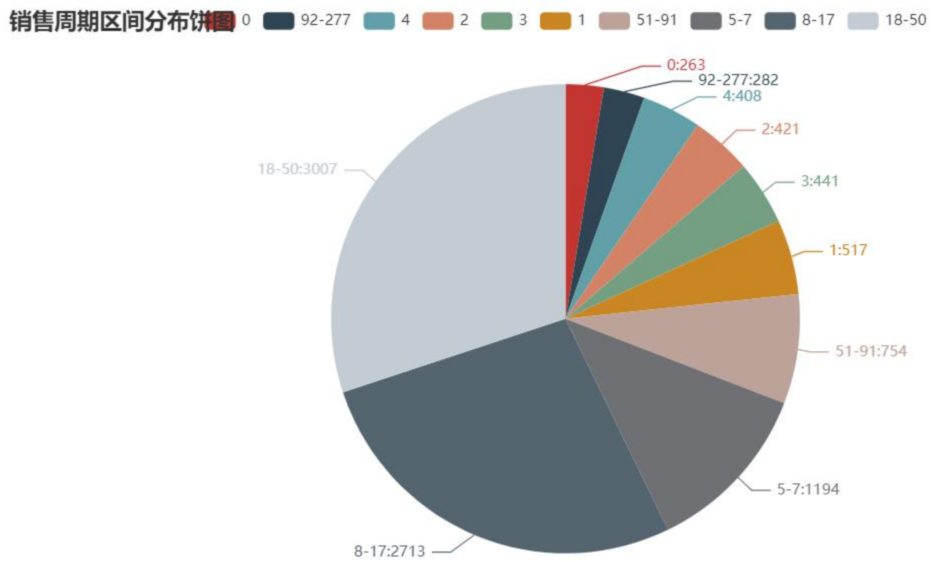


图 18 销售周期区间分布情况

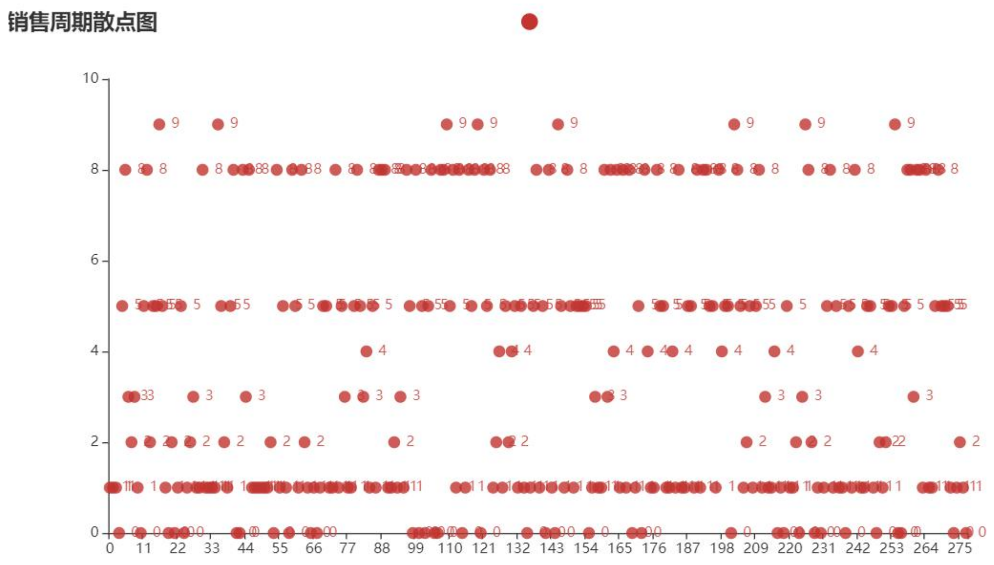


图 19 整体的销售周期区间的分布情况

其次，我们参考了品牌对销售周期的影响，绘制销售量前十的品牌的销售周期区间的变化

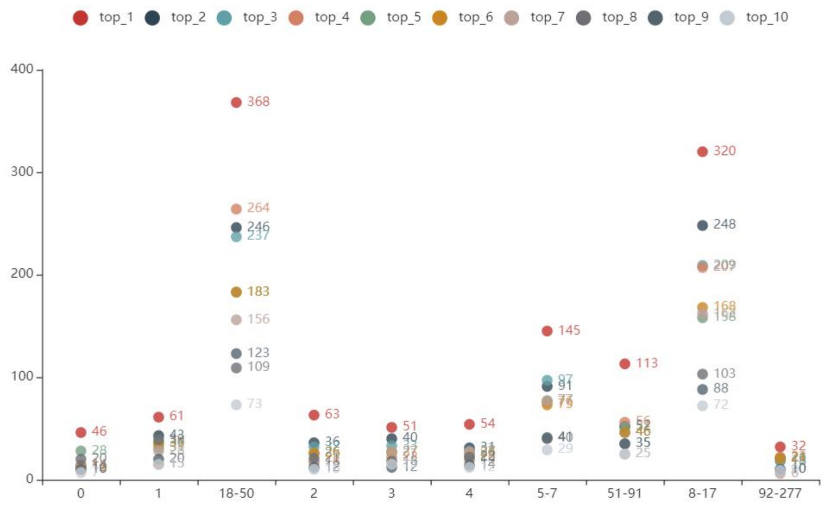


图 20 品牌销量前十的销售区间的数量分布

从图中，可以看出，18-50 与 8-17 的区间中销售量较高。

之后，我们还进行了车辆颜色、过户次数、载客人数、燃油类型等数据分类清新的数据进行可观察，依次查看对二手车销量情况的分析。

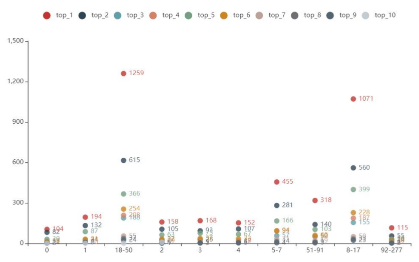


图 21 车辆颜色相应区间数量分析

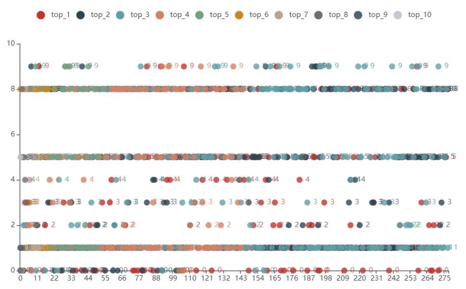


图 22 过户次数对应的整体的销售周期区间分布情况

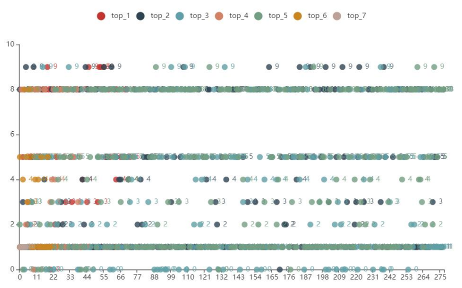


图 23 载客人数对应的整体的销售周期区间分布情况

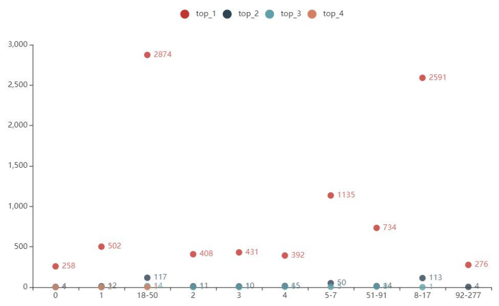


图 24 燃油类型相应区间数量分析

从图中可以看出，车辆颜色、过户次数、载客人数、燃油类型这些数据对二手车的销量周期存在影响，但影响性不大，相应的情况回造成二手车的销量改变，但无法对二手车的整体销量周期造成影响。

6.3 影响因素进一步挖掘

6.3.1 方法与思路说明

1. **分析思路**：利用特征排序（特征选择）[16]对成交周期进行分析与模型构建，目的在于减少特征数量、降维，使得模型的泛化能力更强，减少过拟合。除此之外，考虑到特征值的实际含义，还可以增强对特征和特征值之间的理解。

2. **分析方法**：借助单变量特征选择中的 Pearson 相关系数进行分析，单变量特征选择能够对每一个特征进行测试，处理特征和响应变量之间的关系，根据得分舍弃掉不好的特征，从而衡量变量之间的线性关系。该方法通常对于理解数据具有较好的效果[17]。

6.3.2 基于皮尔逊相关系数的模型架构

1.**数据处理**：将附件 4 和附件 1 进行合并，即通过数据整合得到全面特征与对应特征值。

2. **Pearson 相关系数计算**

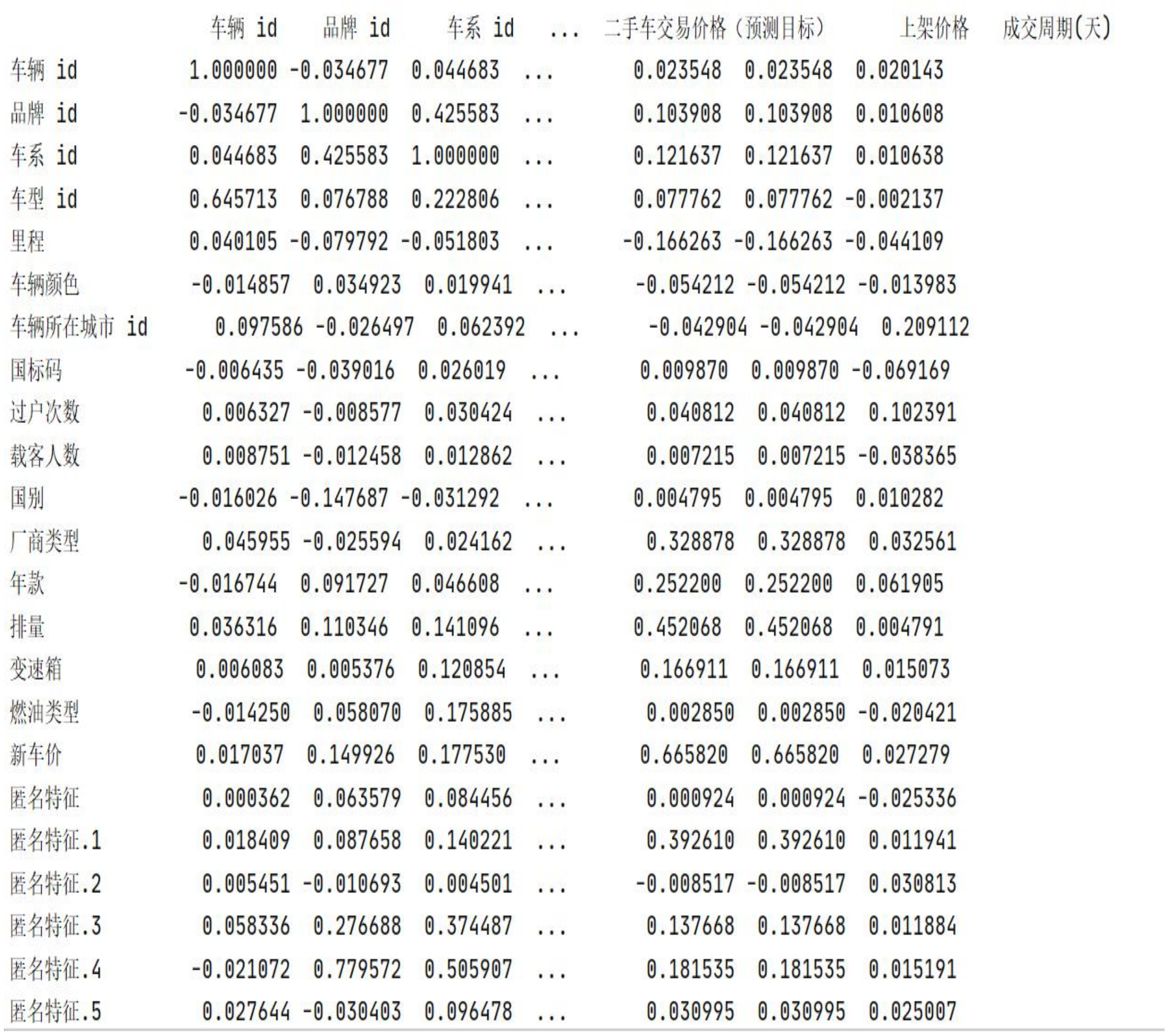
皮尔森相关系数（Pearson correlation coefficient）也称皮尔森积矩相关系数(Pearson product-moment correlation coefficient) ，是一种线性相关系数。皮尔森相关系数是用来反映两个变量线性相关程度的统计量。

借助 pandas 进行相关系数的计算，主要代码如下：

```
data = pd.read_csv(r"C:\Users\Lenovo\Desktop\总.csv")
df = pd.DataFrame(data)
```

```
print(df.corr())
#print(df.corr('spearman'))
#print(df.corr('kendall'))
```

通过计算得到的部分结果如下所示：



```
                车辆 id    品牌 id    车系 id   ...  二手车交易价格（预测目标）   上架价格   成交周期(天)
车辆 id        1.000000 -0.034677  0.044683  ...       0.023548  0.023548  0.020143
品牌 id       -0.034677  1.000000  0.425583  ...       0.103908  0.103908  0.010608
车系 id        0.044683  0.425583  1.000000  ...       0.121637  0.121637  0.010638
车型 id        0.645713  0.076788  0.222806  ...       0.077762  0.077762 -0.002137
里程           0.040105 -0.079792 -0.051803  ...      -0.166263 -0.166263 -0.044109
车辆颜色       -0.014857  0.034923  0.019941  ...      -0.054212 -0.054212 -0.013983
车辆所在城市 id   0.097586 -0.026497  0.062392  ...      -0.042904 -0.042904  0.209112
国标码        -0.006435 -0.039016  0.026019  ...       0.009870  0.009870 -0.069169
过户次数        0.006327 -0.008577  0.030424  ...       0.040812  0.040812  0.102391
载客人数        0.008751 -0.012458  0.012862  ...       0.007215  0.007215 -0.038365
国别          -0.016026 -0.147687 -0.031292  ...       0.004795  0.004795  0.010282
厂商类型        0.045955 -0.025594  0.024162  ...       0.328878  0.328878  0.032561
年款          -0.016744  0.091727  0.046608  ...       0.252200  0.252200  0.061905
排量           0.036316  0.110346  0.141096  ...       0.452068  0.452068  0.004791
变速箱          0.006083  0.005376  0.120854  ...       0.166911  0.166911  0.015073
燃油类型       -0.014250  0.058070  0.175885  ...       0.002850  0.002850 -0.020421
新车价          0.017037  0.149926  0.177530  ...       0.665820  0.665820  0.027279
匿名特征        0.000362  0.063579  0.084456  ...       0.000924  0.000924 -0.025336
匿名特征.1      0.018409  0.087658  0.140221  ...       0.392610  0.392610  0.011941
匿名特征.2      0.005451 -0.010693  0.004501  ...      -0.008517 -0.008517  0.030813
匿名特征.3      0.058336  0.276688  0.374487  ...       0.137668  0.137668  0.011884
匿名特征.4     -0.021072  0.779572  0.505907  ...       0.181535  0.181535  0.015191
匿名特征.5      0.027644 -0.030403  0.096478  ...       0.030995  0.030995  0.025007
```

根据以上得分舍弃掉不好的特征，从而衡量变量之间的线性关系，为进一步分析提供有力依据。其中，对交易周期影响因素从大到小的影响为：

影响因素	相关系数
成交周期(天)	1.0
车辆所在城市 id	0.23534563229055183
过户次数	0.10216949426423909
日期差	0.08962993102729372
上架价格	0.0707070365524042
二手车交易价格（预测目标）	0.06494542929683957
国标码	0.06431636303629248
年款	0.060578636444364176
厂商类型	0.0487196933921786
匿名特征.12	0.04655001366089112
里程	0.03478575214638492

匿名特征.7	0.028141715098782813
匿名特征.13	0.027435873114480064
匿名特征.9	0.026132094159409053
载客人数	0.024889749547639145
新车价	0.023500389301353265
匿名特征.2	0.022423140276252595
车系 id	0.022367014670937595
价格差	0.02167497669814242
匿名特征.5	0.021066990055458513
匿名特征	0.020783016045806048
车辆 id	0.018827544991527216
匿名特征.1	0.015929103972915033
品牌 id	0.014789694600914423
匿名特征.4	0.014051087602147447
排量	0.009639581177829268
变速箱	0.009395586940910267
车辆颜色	0.007567743214531442
国别	0.004328849560166613
匿名特征.8	0.003765964452441954
燃油类型	0.002944137358971221
匿名特征.3	0.0015831774038671997
车型 id	0.0005102795570690364

表 1 相关系数分析表

3.Pearson 相关系数绘图展示与分析

根据以上计算，绘制相关系数图如下所示：

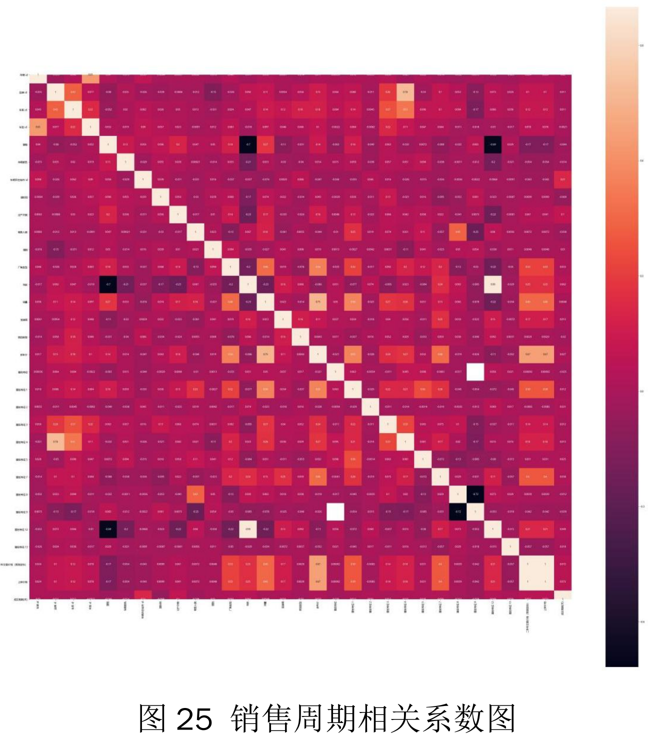


图 25 销售周期相关系数图

依据相关系数图可以看出，成交价格仍然是影响成交周期强有力的因素。

再进一步分析过程中，注意到价格和日期对成交周期的影响因素较大，其中价格对交易周期的相关系数为 0.0707070365524042，在对日期进行分析时，考虑到日期的不同，进行日期差进行计算，其中日期差 = 上架时间 – 注册日期。对日期差进行相关系数分析，得到的相关系数为 0.0896299310272937。对比绘图展示，如下图：

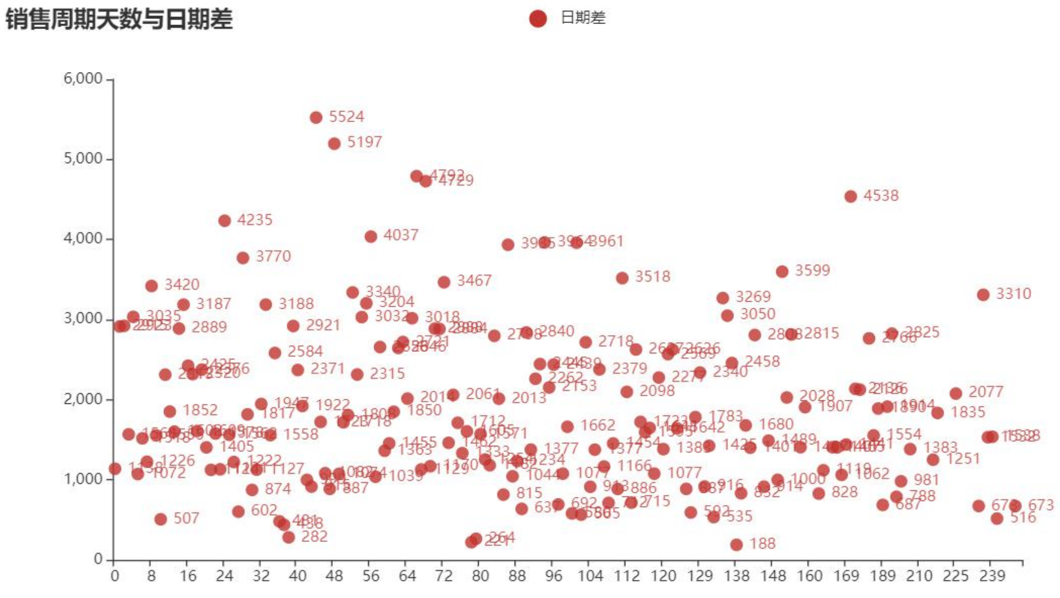


图 26 交易周期与日期差

从图中可以看出，日期差较小的，其交易周期也较短，可以更好的进行二手车的出售。

6.4 采用关键因素加快在库车辆的销售速度

需要加快门店在库车辆的销售速度，本报告欲结合以上分析得出的关键因素采取对应的手段，并说明这些手段的适用条件和预期效果。

6.4.1 基于价格因素加快在库车辆销售速度

由问题二的分析、挖掘、皮尔逊相关系数计算等措施，得出影响二手车成交周期的最大关键因素为价格。因而上架的二手车如何定价显然成为了关键问题。

故而基于价格进行在库车辆销售速度加快策略分析：

（1）由于二手车的定价要依托于新车市场的行情与售价，故而不能以卖家期望盈利值去定价，应当注意行市变化。注意该种车辆的新车价格，由该价格确定合适且能够最大限度满足盈利预期的二手车定价，自然会有消费者在权衡利弊之下倾向于二手车。

（2）依据二手车上架后买家的反馈，要及时调整定价，常常更新以达到盈利最大化。对于迟迟未卖出的二手车而言，降低价格时候采取多次少量的手段，即不能一次性降到利润最小值对应的最低价格，而是以一定价格步长一次次逼近

最低价格。这样不至于利益直接跌到最小。

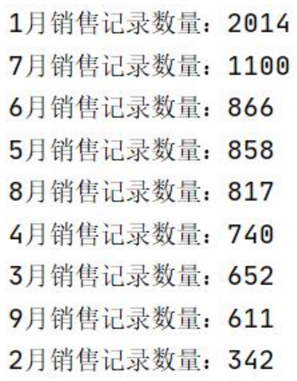
七、问题三的探究

依据所给样本数据集，下文给出本报告认为值得研究的问题与相关思路。

7.1 淡旺季之于二手车买卖的影响探讨

经统计发现二手车买卖存在淡旺季，卖家一般都会选择在年底清货，避免下一年该车贬值。由于二手车价值是按年来折算的，12 月和 1 月即使仅仅相差一个月，虽然就实际性能来讲，相差不大，但对于车龄来说，的确相差一年。虽然依据现有数据并无 12 月销售情况，但可以推测出 12 月很大程度上可以算是二手车买卖的旺季。另外一方面，到了年底，买车的也会增多，对二手车的需求会增大，这也是是导致年底成为旺季的原因之一。除此之外，夏季（暑假）也可以看做是二手车买卖旺季，二手车价格最低的时候，应该是在最炎热的 7 月下旬左右，这个时候因为天气炎热，很少有人去买二手车，车商为了促进交易一般会给予一定的优惠，所以这个时候买二手车是最划算的，想购买二手车的也会选择这个时候。

根据现有数据，得到统计结果如下：


```
1月销售记录数量：2014
7月销售记录数量：1100
6月销售记录数量：866
5月销售记录数量：858
8月销售记录数量：817
4月销售记录数量：740
3月销售记录数量：652
9月销售记录数量：611
2月销售记录数量：342
```

依据以上情况分析，淡旺季之于二手车买卖影响不可小觑。

由以上分析，本报告给出针对淡旺季情况的增加盈利的思路：由历史记录数据不仅要得出二手车全行业的淡旺季，而且要根据不同车型等车辆特征得出对应的“基于不同车辆特征本身的淡旺季”。从而在旺季增加该种车辆的库存，达到盈利的目的。淡季可以采取降价促销、优惠活动等措施清理库存，保证不亏本。

7.2 处置积压库存车辆策略分析

在二手车销售的过程中，必须保证库存的新鲜度，理论上说成交周期越短越好。二手车受到新车促销降价影响很大，有很强的季节性风险，有明显的淡旺季之分。需要在合适的时间压缩库存，处置积压车辆。也就是说，二手车卖家并不是全年盈利，而是旺季盈利，淡季保本（保证不亏本）。

由以上分析，本报告给出针对处置积压库存车辆以增加盈利的思路：针对积压车辆带来的问题，可以采取旺季提高利润百分点，以达到弥补淡季亏本的目的。

7.3 价格差带来的利润问题

在二手车买卖的过程中，鉴于市场或卖家需求、该种车辆存货积压等问题，存在更新价格的情况。需求增大时，抬高价格以达到盈利目的，但是当该车辆长期卖不出去时，需要降价处理。那么降价的“度”是本报告要探讨的。在急于处理车辆的同时，不可以一味以清货为目的，还要考虑到是否盈利的问题。故而变更价格时，针对降价的车辆，要衡量利润，即在车辆上架时候就可以根据以往销售记录确定该车辆最低价格，达到利益最大化的效果。即对于根据二手车的新旧程度，控制二手车的价格差，不可一味只顾赚钱，不管车辆的价格空间。

由以上分析，本报告给出价格差带来的利润问题的解决思路：在二手车的销售过程中，考虑到整体的销售情况，为保证车辆的合理销售，应当在车辆上架之前依据二手车卖家的盈利预期计算该批车辆利润，确定在变更价格的过程中的最低车价。

八、模型检验

8.1 问题一模型

将训练集拆分后根据预测值和实际的值绘制散点图进行观察

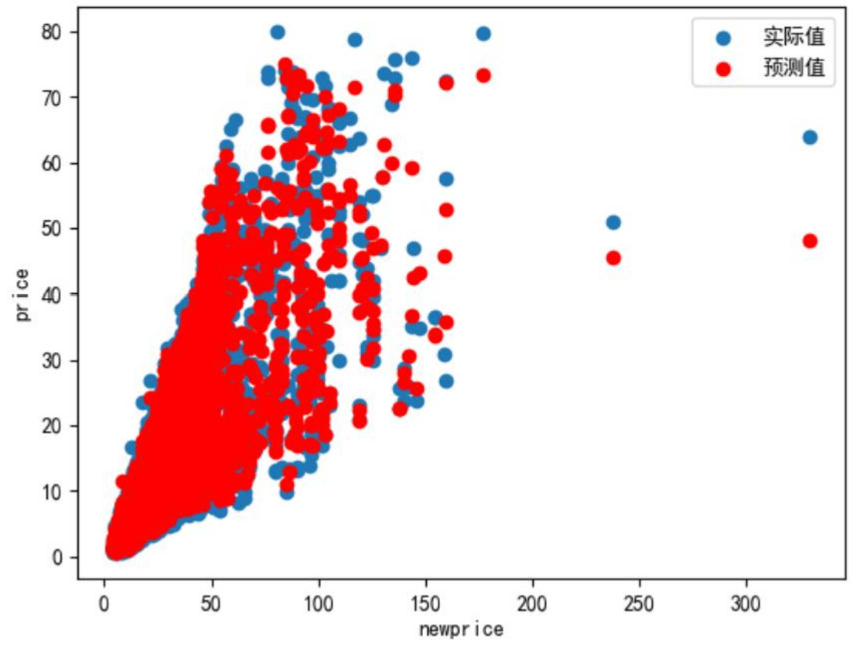


图 27 预测值和实际的值绘制散点图

8.2 问题二模型

将预测值改为周期，进行回归预测

```
Out[68]: 0.9000438912072556
```

得分最高的时候选择的特征就是对周期影响因素最大的特征

九、模型评价

9.1 问题一模型的优点和缺点

问题一基于随机森林模型，随机森林是一个包含多个决策树的分类器。

1. 优点：

（1）针对较大数据集，有着较好的预测效果。

（2）训练完模型之后可以直接得出关键特征，即最具影响力的一些特征集合。

（3）在创建随机森林的时候，对 generlization error 使用的是无偏估计，因此模型泛化能力强。

（4）因为树与树之间是相互独立的，故训练速度相对较快。

（5）实现较为容易，预测过程也并不复杂。

2. 缺点：

（1）当进行回归时，若超越训练集数据范围的预测，这可能导致出现过度拟合的情况。

（2）无法进入模型内部，只能通过调参和随机种子得到更好的效果。

9.2 问题二模型分析

采用与问题一相同的模型，基于 xgboost 可解决过拟合、能够减少计算的优点，之后利用 xgboost 进行模型优化。进一步提升模型的性能。

参考文献

[1] 冯秀荣, 王斌. 影响二手车价值的因子分析[J]. 商业研究, 2008(2):4.
[2] 丁礼灯, 席敏. 我国二手车市场现状分析[J]. 法制与社会：锐视版, 2008(24):2.
[3] 蔡云, 唐岚, 谭金会. 中国二手车市场的发展分析[J]. 中国集体经济, 2009(10S):2.
[4] 吴勃生, 郭殿臣. 我国二手车市场的发展状况及经营策略调整[J]. 中国科技信息, 2010(22):2.
[5] 杨凯, 王军雷, 康凯. 国内二手车市场现状与展望[J]. 汽车工业研究, 2014.
[6] 栾兆宇. 中国品牌二手车市场营销模式研究[D]. 吉林大学, 2011.
[7] 何继志. 二手车市场发展的现状与前景展望[J]. 中国工程咨询, 2003(2):3.
[8] 傅志中, 吴宇峰, 徐进,等. 一种基于特征处理的图像信息融合及超分辨率重建方法:, CN112598575A[P]. 2021.
[9] 陈功平, 王红. 改进 Pearson 相关系数的个性化推荐算法[J]. 山东农业大学学报：自然科学版, 2016, 47(6):5.
[10] 李靖华, 郭耀煌. 主成分分析用于多指标评价的方法研究——主成分评价[J]. 管理工程学报, 2002, 16(1):5.
[11] 吴成茂. 基于加权香农熵的图像阈值法[J]. 计算机工程与应用, 2008, 44(18):4.
[12] 克劳德 E. 香农. 《通信的数学原理》. 《贝尔实验室技术杂志》.1948
[13] Chen T , Tong H , Benesty M . xgboost: Extreme Gradient Boosting[J]. 2016.S
[14] 佚名. 二手车销售[J]. 汽车驾驶员, 2006(2):2.
[15] 杨斌. 二手车销售状况与发展趋势研究[J]. 湖北农机化, 2019(10):1.
[16] 邓貌. 模式识别中特征处理方法的应用研究[J]. 2009.
[17] 于玲, 吴铁军. 集成学习:Boosting 算法综述[J]. 模式识别与人工智能, 2004, 17(1):8.

附录

使用软件名称：PyCharm

附录 1：问题一代码

1.模型训练

```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from category_encoders.leave_one_out import LeaveOneOutEncoder
from sklearn.model_selection import cross_val_score
from sklearn.decomposition import PCA
import xgboost as xgb

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
"""
评估函数
"""
def estimate(y_true=None,y_pred=None):
    y_true=np.array(list(y_true))
    y_pred=np.array(list(y_pred))
    Ape=np.abs(y_pred-y_true)/y_true
    Mape=sum(Ape)/len(y_true)
    # Ape_count=[np.nan if x <=0.05 else x for x in Ape]
    Ape_count=len(np.where(Ape<=0.05)[0])/len(Ape)
    return 0.2*(1-Mape)+0.8*Ape_count

"""
读入数据和数据清洗
"""
```

```
df = pd.read_table(r'D:\比赛\2021年MathorCup大数据竞赛赛道A\附件\附件1：
估价训练数据.txt',
                         parse_dates=[1, 11, 12],sep='\t', header=None,
encoding='gbk')
# df = pd.read_table('../data/附件 1：估价训练数据.txt', sep='\t', header=None,
encoding='gbk')
data = pd.DataFrame(data=df)
columns = ['carid', 'tradeTime', 'brand', 'serial', 'model', 'mileage', 'color', 'cityId',
'carCode',
           'transferCount', 'seatings', 'registerDate', 'licenseDate', 'country',
'maketype', 'modelyear',
           'displacement', 'gearbox', 'oiltype', 'newprice']
for i in range(1, 16):
    str_ = 'anonymousFeature'+str(i)
    columns.append(str_)
columns.append('price')
data.columns = columns

data=data.drop(['country','anonymousFeature4','anonymousFeature7','anonymous
Feature10','anonymousFeature15'
                ,'maketype','anonymousFeature1','anonymousFeature8','anony
mousFeature9'],axis=1) #  删除缺失值严重的特征
data.info()
nn = ['carCode', 'modelyear', 'gearbox']

for i in nn:
    x = int(data[i].mode())
    data[i].fillna(x, inplace=True)

"""
特征构造
"""
data['tradeTime_year']=data['tradeTime'].dt.year
data['tradeTime_month']=data['tradeTime'].dt.month
data['tradeTime_day']=data['tradeTime'].dt.day
data['registerDate_year']=data['registerDate'].dt.year
data['registerDate_month']=data['registerDate'].dt.month
data['registerDate_day']=data['registerDate'].dt.day

print(data['registerDate_day'].value_counts().plot(kind='bar',rot=30))
plt.show()

#  处理定类数据（Frequency 编码）
data['color'] = data['color'].map(data['color'].value_counts())
```

```
data['carCode'] = data['carCode'].map(data['carCode'].value_counts())
data['modelyear'] = data['modelyear'].map(data['modelyear'].value_counts())

# 处理匿名特征 11
data_1=data.dropna()                          #
data_1.info()

def deal_11(x):
     return sum([float(x) for x in re.findall("\d",x)])
data_1['anonymousFeature11']=data_1['anonymousFeature11'].map(deal_11)
data_1.info()
# 处理匿名特征 12
def deal_12(x):
     li=[float(x) for x in re.findall("\d+",x)]
     return li[0]*li[1]*li[2]
data_1['anonymousFeature12_length']=data_1['anonymousFeature12'].apply(lam
bda x:int(x.split('*')[0]))
data_1['anonymousFeature12_width']=data_1['anonymousFeature12'].apply(lam
bda x:int(x.split('*')[1]))
data_1['anonymousFeature12_height']=data_1['anonymousFeature12'].apply(lam
bda x:int(x.split('*')[2]))
data_1['anonymousFeature12']=data_1['anonymousFeature12'].map(deal_12)
# 处理匿名特征 13
def deal_13(x):
     return x[:4], x[4:6]
data_1['anonymousFeature13']=data_1['anonymousFeature13'].astype('string')
data_1['anonymousFeature13_year']=data_1['anonymousFeature13'].map(deal_1
3)     #   (2017, 09)
data_1['anonymousFeature13_month']=data_1['anonymousFeature13_year'].appl
y(lambda x: int(x[1]))
data_1['anonymousFeature13_year']=data_1['anonymousFeature13_year'].apply(l
ambda x: int(x[0]))
data_1['anonymousFeature13']=data_1['anonymousFeature13'].astype('float')


# 构建新特征
data_1['old_year']=data_1['tradeTime']-data_1['registerDate']
data_1['old_year']=data_1['old_year'].apply(lambda x:str(x).split(' ')[0])
data_1['old_year']=data_1['old_year'].astype(int)

data_1['old_year_1']=data_1['tradeTime']-data_1['licenseDate']
data_1['old_year_1']=data_1['old_year_1'].apply(lambda x:str(x).split(' ')[0])
data_1['old_year_1']=data_1['old_year_1'].astype(int)
```

```
# 数据分桶
bin=[0, 1, 4, 7.15, 10, 50]
data_1['mileage_bin']=pd.cut(data_1['mileage'],bins=bin,labels=False)

y = data_1['price']
data_2 = data_1.drop(['price'],axis=1)
print(data_2.columns)
data_2['price'] = y
data_2.info()
# data_2=data_2.dropna()
data_2=data_2.drop(['tradeTime','registerDate','licenseDate'],axis=1)

# 看看有没有异常值
data_2.describe()
data_2=data_2[data_2['price']<80]
data_2.columns
data_2.info()

"""
特征选择
"""
# 基于皮尔逊相关系数
pearson = data_2.corr()
index = pearson['price'][:-1].abs() > 0.1
X = data_2.iloc[:,:-1]
X_subset = X.loc[:, index]
# X_subset.columns

"""
降维
"""
# pca=PCA(n_components=6)
# X_new_sk=pca.fit_transform(X_subset)


"""
模型训练
"""
# standardScaler=StandardScaler()
# X = standardScaler.fit_transform(X_subset)

X_train, X_test, y_train, y_test =
train_test_split(X_subset,data_2['price'].to_numpy() ,
test_size=0.2,random_state=3)

random_model =
RandomForestRegressor(n_estimators=500,random_state=33,n_jobs=-1)
random_model.fit(X_train,y_train)
y_pred = random_model.predict(X_test)
score = estimate(y_true=y_test,y_pred=y_pred)

"""
XGboost 优化模型
"""
xlf = xgb.XGBRegressor(max_depth=50,   # 差不多
                                learning_rate=0.05,
                                n_estimators=300,
                                min_child_weight=2,
                                subsample=0.8,
                                colsample_bytree=0.8,
                                seed=0,
                                )
xlf.fit(X_train, y_train, eval_metric='mae', eval_set=[(X_train, y_train), (X_test,
y_test)], early_stopping_rounds=20)
preds = xlf.predict(X_test)
score=estimate(y_true=y_test,y_pred=preds)
score

"""
画散点图观察模型的效果
"""
random_model.score(X_test,y_test)
random_model.score(X_train,y_train)
np.array(y_test)
plt.scatter(X_test['newprice'], y_test)   # 样本实际分布
plt.scatter(X_test['newprice'], y_pred, color='red')     # 绘制拟合曲线
plt.legend(['实际值','预测值'])
plt.xlabel("newprice")
plt.ylabel("price")
plt.show()
```

2.模型预测

```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from category_encoders.leave_one_out import LeaveOneOutEncoder
from sklearn.model_selection import cross_val_score
import xgboost as xgb

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

def estimate(y_true=None,y_pred=None):
    y_true=np.array(list(y_true))
    y_pred=np.array(list(y_pred))
    Ape=np.abs(y_pred-y_true)/y_true
    Mape=sum(Ape)/len(y_true)
    # Ape_count=[np.nan if x <=0.05 else x for x in Ape]
    Ape_count=len(np.where(Ape<=0.05)[0])/len(Ape)
    return 0.2*(1-Mape)+0.8*Ape_count


def train(data_0=None):
    data=pd.DataFrame(data=data_0)
    columns = ['carid', 'tradeTime', 'brand', 'serial', 'model', 'mileage', 'color',
'cityId', 'carCode',
               'transferCount', 'seatings', 'registerDate', 'licenseDate', 'country',
'maketype', 'modelyear',
               'displacement', 'gearbox', 'oiltype', 'newprice']
    for i in range(1, 16):
        str_ = 'anonymousFeature' + str(i)
        columns.append(str_)
    columns.append('price')
    data.columns = columns

    #
data=data.drop(['anonymousFeature4','anonymousFeature7','anonymousFeature1
0','anonymousFeature15'],axis=1)   #  删除缺失值严重的特征
    data = data.drop(['country', 'anonymousFeature4', 'anonymousFeature7',
'anonymousFeature10', 'anonymousFeature15'
                          , 'maketype', 'anonymousFeature1',
'anonymousFeature8', 'anonymousFeature9'], axis=1)
    data.info()
    nn = ['carCode', 'modelyear', 'gearbox']
```

```
for i in nn:
    x = int(data[i].mode())
    data[i].fillna(x, inplace=True)


"""
特征构造
"""
data['tradeTime_year'] = data['tradeTime'].dt.year
data['tradeTime_month'] = data['tradeTime'].dt.month
data['tradeTime_day'] = data['tradeTime'].dt.day
data['registerDate_year'] = data['registerDate'].dt.year
data['registerDate_month'] = data['registerDate'].dt.month
data['registerDate_day'] = data['registerDate'].dt.month

print(data['registerDate_day'].value_counts().plot(kind='bar', rot=30))
plt.show()

# 处理定类数据（Frequency 编码）
data['color'] = data['color'].map(data['color'].value_counts())
data['carCode'] = data['carCode'].map(data['carCode'].value_counts())
# data['country'] = data['country'].map(data['country'].value_counts())
data['modelyear'] = data['modelyear'].map(data['modelyear'].value_counts())

# 处理匿名特征 11
data_1 = data.dropna()   #
data_1.info()
len(list(data_1['anonymousFeature11']))


def deal_11(x):
    return sum([float(x) for x in re.findall("\d", x)])

data_1['anonymousFeature11'] =
data_1['anonymousFeature11'].map(deal_11)
data_1.info()

# 处理匿名特征 12
def deal_12(x):
    li = [float(x) for x in re.findall("\d+", x)]
    return li[0] * li[1] * li[2]

data_1['anonymousFeature12_length'] =
data_1['anonymousFeature12'].apply(lambda x: int(x.split('*')[0]))
    data_1['anonymousFeature12_width'] =
data_1['anonymousFeature12'].apply(lambda x: int(x.split('*')[1]))
    data_1['anonymousFeature12_height'] =
data_1['anonymousFeature12'].apply(lambda x: int(x.split('*')[2]))
    data_1['anonymousFeature12'] =
data_1['anonymousFeature12'].map(deal_12)

    # 处理匿名特征 13
    def deal_13(x):
        return x[:4], x[4:6]

    data_1['anonymousFeature13'] =
data_1['anonymousFeature13'].astype('string')
    data_1['anonymousFeature13_year'] =
data_1['anonymousFeature13'].map(deal_13)   # (2017, 09)
    data_1['anonymousFeature13_month'] =
data_1['anonymousFeature13_year'].apply(lambda x: int(x[1]))
    data_1['anonymousFeature13_year'] =
data_1['anonymousFeature13_year'].apply(lambda x: int(x[0]))
    data_1['anonymousFeature13'] =
data_1['anonymousFeature13'].astype('float')


    data_1['old_year'] = data_1['tradeTime'] - data_1['registerDate']
    data_1['old_year'] = data_1['old_year'].apply(lambda x: str(x).split(' ')[0])
    data_1['old_year'] = data_1['old_year'].astype(int)

    data_1['old_year_1'] = data_1['tradeTime'] - data_1['licenseDate']
    data_1['old_year_1'] = data_1['old_year_1'].apply(lambda x: str(x).split(' ')[0])
    data_1['old_year_1'] = data_1['old_year_1'].astype(int)


    # 数据分桶
    bin = [0, 1, 4, 7.15, 10, 50]
    data_1['mileage_bin'] = pd.cut(data_1['mileage'], bins=bin, labels=False)

    y = data_1['price']
    data_2 = data_1.drop(['price'], axis=1)
    print(data_2.columns)
    data_2['price'] = y
    data_2.info()
    # data_2=data_2.dropna()
    data_2 = data_2.drop(['tradeTime', 'registerDate', 'licenseDate'], axis=1)
```

```
        # 看看有没有异常值
        data_2.describe()
        data_2 = data_2[data_2['price'] < 80]
        data_2.columns
        data_2.info()
        # data_2.drop([22115],axis=0,inplace=True)
        """
        特征选择
        """
        # 基于皮尔逊相关系数
        pearson = data_2.corr()
        index = pearson['price'][:-1].abs() > 0.1
        X = data_2.iloc[:, :-1]
        X_subset = X.loc[:, index]
        # X_subset.columns

        """
        降维
        """

        """
        模型训练
        """
        # standardScaler = StandardScaler()
        # X = standardScaler.fit_transform(X_subset.to_numpy())

        X_train, X_test, y_train, y_test = train_test_split(X_subset,
data_2['price'].to_numpy(), test_size=0.2, random_state=3)

        # random_model = RandomForestRegressor(n_estimators=500,
random_state=33, n_jobs=-1)
        # random_model.fit(X_train, y_train)
        # y_pred = random_model.predict(X_test)
        xlf = xgb.XGBRegressor(max_depth=50,   # 差不多
                                learning_rate=0.05,
                                n_estimators=300,
                                min_child_weight=2,
                                subsample=0.8,
                                colsample_bytree=0.8,
                                seed=0,
                                )
        xlf.fit(X_train, y_train, eval_metric='mae', eval_set=[(X_train, y_train), (X_test,
y_test)],
                    early_stopping_rounds=20)
          preds = xlf.predict(X_test)
          return xlf,X_subset.columns,preds,y_test
def test(data_0=None,model=None,feature=None):
          data=pd.DataFrame(data=data_0)
          columns = ['carid', 'tradeTime', 'brand', 'serial', 'model', 'mileage', 'color',
'cityId', 'carCode',
                                   'transferCount', 'seatings', 'registerDate', 'licenseDate', 'country',
'maketype', 'modelyear',
                                   'displacement', 'gearbox', 'oiltype', 'newprice']
          for i in range(1, 16):
                    str_ = 'anonymousFeature' + str(i)
                    columns.append(str_)
          data.columns = columns


          #
data=data.drop(['anonymousFeature4','anonymousFeature7','anonymousFeature1
0','anonymousFeature15'],axis=1)    #  删除缺失值严重的特征
          data = data.drop(['country', 'anonymousFeature4', 'anonymousFeature7',
'anonymousFeature10', 'anonymousFeature15'
                                                          , 'maketype', 'anonymousFeature1',
'anonymousFeature8', 'anonymousFeature9'], axis=1)

          nn = ['carCode', 'modelyear', 'gearbox','anonymousFeature13']
          for i in nn:
                    x = int(data[i].mode())
                    data[i].fillna(x, inplace=True)


          """
          特征构造
          """
          data['tradeTime_year'] = data['tradeTime'].dt.year
          data['tradeTime_month'] = data['tradeTime'].dt.month
          data['tradeTime_day'] = data['tradeTime'].dt.day
          data['registerDate_year'] = data['registerDate'].dt.year
          data['registerDate_month'] = data['registerDate'].dt.month
          data['registerDate_day'] = data['registerDate'].dt.day



          #  处理定类数据（Frequency 编码）
          data['color'] = data['color'].map(data['color'].value_counts())
```

```
    data['carCode'] = data['carCode'].map(data['carCode'].value_counts())
    # data['country'] = data['country'].map(data['country'].value_counts())
    data['modelyear'] = data['modelyear'].map(data['modelyear'].value_counts())

    # 处理匿名特征 11
    data_1=data
    data_1.info()
    data_1['anonymousFeature11'].fillna('3',inplace=True)
    def deal_11(x):
        return sum([float(x) for x in re.findall("\d", x)])

    data_1['anonymousFeature11'] =
data_1['anonymousFeature11'].map(deal_11)
    data_1.info()

    # 处理匿名特征 12
    data_1['anonymousFeature12'].fillna('5015*1874*1455',inplace=True)
    def deal_12(x):
        li = [float(x) for x in re.findall("\d+", x)]
        return li[0] * li[1] * li[2]

    data_1['anonymousFeature12_length'] =
data_1['anonymousFeature12'].apply(lambda x: int(x.split('*')[0]))
    data_1['anonymousFeature12_width'] =
data_1['anonymousFeature12'].apply(lambda x: int(x.split('*')[1]))
    data_1['anonymousFeature12_height'] =
data_1['anonymousFeature12'].apply(lambda x: int(x.split('*')[2]))
    data_1['anonymousFeature12'] =
data_1['anonymousFeature12'].map(deal_12)

    # 处理匿名特征 13
    def deal_13(x):
        return x[:4], x[4:6]

    data_1['anonymousFeature13'] =
data_1['anonymousFeature13'].astype('string')
    data_1['anonymousFeature13_year'] =
data_1['anonymousFeature13'].map(deal_13)    # (2017, 09)
    data_1['anonymousFeature13_month'] =
data_1['anonymousFeature13_year'].apply(lambda x: int(x[1]))
    data_1['anonymousFeature13_year'] =
data_1['anonymousFeature13_year'].apply(lambda x: int(x[0]))
    data_1['anonymousFeature13'] =
data_1['anonymousFeature13'].astype('float')
```

```
    data_1['old_year'] = data_1['tradeTime'] - data_1['registerDate']
    data_1['old_year'] = data_1['old_year'].apply(lambda x: str(x).split(' ')[0])
    data_1['old_year'] = data_1['old_year'].astype(int)

    data_1['old_year_1'] = data_1['tradeTime'] - data_1['licenseDate']
    data_1['old_year_1'] = data_1['old_year_1'].apply(lambda x: str(x).split(' ')[0])
    data_1['old_year_1'] = data_1['old_year_1'].astype(int)



    # 数据分桶
    bin = [0, 1, 4, 7.15, 10, 50]
    data_1['mileage_bin'] = pd.cut(data_1['mileage'], bins=bin, labels=False)
    data_2 = data_1.drop(['tradeTime', 'registerDate', 'licenseDate'], axis=1)
    X=data_2[feature]
    y=model.predict(X)
    return X ,y

df = pd.read_table(r'D:\比赛\2021 年 MathorCup 大数据竞赛赛道 A\附件\附件 1：
估价训练数据.txt',
                        parse_dates=[1, 11, 12],sep='\t', header=None,
encoding='gbk')
df_1 = pd.read_table(r'D:\比赛\2021 年 MathorCup 大数据竞赛赛道 A\附件\附件
2：估价验证数据.txt',
                        parse_dates=[1, 11, 12],sep='\t', header=None,
encoding='gbk')

model,feature,y_p,y_t=train(data_0=df)
score=estimate(y_true=y_t,y_pred=y_p)
X, y_=test(data_0=df_1,model=model,feature=feature)
data_=pd.DataFrame(df_1.iloc[:,0])
data_['price']=y_
print(data_)
data_.to_csv('../data/附件 3：估价模型结果.txt',sep='\t',index=False,
header=None)
# data_.to_csv(r'D:\比赛\2021 年 MathorCup 大数据竞赛赛道 A\附件 3：估价模
型结果.txt',sep='\t',index=False, header=None)
```

附录 2：问题二代码

2.1 结合附件四

2.1.1 初步分析

```
import pandas as pd
from collections import Counter
data = pd.read_csv("C:\\Users\Lenovo\Desktop\\4.csv", encoding='utf-8')
#print(data)
listT1= []
listP1= []
listT2= []
listP2= []
listT3= []
listP3= []
listT4= []
listP4= []
listT5= []
listP5= []
listT6= []
listP6= []
listT7= []
listP7= []
listT8= []
listP8= []
listT12= []
listP12= []
list_len = []
for i in data.updatePriceTimeJson:
        i = i.split("")
        list_len.append(len(i))
        if len(i) == 5:#--------------------------------------1
          #print(i)
          #print(len(i))
          listT1.append(i[1])
          listP1.append(i[3])
          x = " "
          #print(i)
          listT2.append(x)
```

```
  listP2.append(x)
  listT3.append(x)
  listP3.append(x)
  listT4.append(x)
  listP4.append(x)
  listT5.append(x)
  listP5.append(x)
  listT6.append(x)
  listP6.append(x)
  listT7.append(x)
  listP7.append(x)
  listT8.append(x)
  listP8.append(x)
  listT12.append(x)
  listP12.append(x)
elif len(i) == 9:#---------------------------------------2
  listT1.append(i[1])
  listP1.append(i[3])
  listT2.append(i[5])
  listP2.append(i[7])
  x = " "
  #print(i)
  listT3.append(x)
  listP3.append(x)
  listT4.append(x)
  listP4.append(x)
  listT5.append(x)
  listP5.append(x)
  listT6.append(x)
  listP6.append(x)
  listT7.append(x)
  listP7.append(x)
  listT8.append(x)
  listP8.append(x)
  listT12.append(x)
  listP12.append(x)
  #print(i[1])
  #data.updateTime = data.updatePriceTimeJson.apply(lambda x:i[1])
  #data.updatePrice = i[3]
elif len(i) == 13:#------------------------------------3
    listT1.append(i[1])
    listP1.append(i[3])
    listT2.append(i[5])
    listP2.append(i[7])
    listT3.append(i[9])
    listP3.append(i[11])
    x = " "
    listT4.append(x)
    listP4.append(x)
    listT5.append(x)
    listP5.append(x)
    listT6.append(x)
    listP6.append(x)
    listT7.append(x)
    listP7.append(x)
    listT8.append(x)
    listP8.append(x)
    listT12.append(x)
    listP12.append(x)
elif len(i) == 17:#-----------------------------------------4
    listT1.append(i[1])
    listP1.append(i[3])
    listT2.append(i[5])
    listP2.append(i[7])
    listT3.append(i[9])
    listP3.append(i[11])
    listT4.append(i[13])
    listP4.append(i[15])
elif len(i) == 21:#-----------------------------------------4
    listT1.append(i[1])
    listP1.append(i[3])
    listT2.append(i[5])
    listP2.append(i[7])
    listT3.append(i[9])
    listP3.append(i[11])
    listT4.append(i[13])
    listP4.append(i[15])
    x = " "
    listP5.append(x)
    listT6.append(x)
    listP6.append(x)
    listT7.append(x)
    listP7.append(x)
    listT8.append(x)
    listP8.append(x)
    listT12.append(x)
    listP12.append(x)
elif len(i) == 25:    # -----------------------------------------5
```

```
    listT1.append(i[1])
    listP1.append(i[3])
    listT2.append(i[5])
    listP2.append(i[7])
    listT3.append(i[9])
    listP3.append(i[11])
    listT4.append(i[13])
    listP4.append(i[15])
    listT5.append(i[17])
    listP5.append(i[19])
    x = " "
    listT6.append(x)
    listP6.append(x)
    listT7.append(x)
    listP7.append(x)
    listT8.append(x)
    listP8.append(x)
    listT12.append(x)
    listP12.append(x)
elif len(i) == 29:    # --------------------------------------6
    listT1.append(i[1])
    listP1.append(i[3])
    listT2.append(i[5])
    listP2.append(i[7])
    listT3.append(i[9])
    listP3.append(i[11])
    listT4.append(i[13])
    listP4.append(i[15])
    listT5.append(i[17])
    listP5.append(i[19])
    listT6.append(i[21])
    listP6.append(i[23])
    x = " "
    listT7.append(x)
    listP7.append(x)
    listT8.append(x)
    listP8.append(x)
    listT12.append(x)
    listP12.append(x)
elif len(i) == 33:    # --------------------------------------7
    listT1.append(i[1])
    listP1.append(i[3])
    listT2.append(i[5])
    listP2.append(i[7])
    listT3.append(i[9])
    listP3.append(i[11])
    listT4.append(i[13])
    listP4.append(i[15])
    listT5.append(i[17])
    listP5.append(i[19])
    listT6.append(i[21])
    listP6.append(i[23])
    listT7.append(i[25])
    listP7.append(i[27])
    x = " "
    listT8.append(x)
    listP8.append(x)
    listT12.append(x)
    listP12.append(x)
elif len(i) == 37:    # ------------------------------------------8
    listT1.append(i[1])
    listP1.append(i[3])
    listT2.append(i[5])
    listP2.append(i[7])
    listT3.append(i[9])
    listP3.append(i[11])
    listT4.append(i[13])
    listP4.append(i[15])
    listT5.append(i[17])
    listP5.append(i[19])
    listT6.append(i[21])
    listP6.append(i[23])
    listT7.append(i[25])
    listP7.append(i[27])
    listT8.append(i[29])
    listP8.append(i[31])
    x = " "
    listT12.append(x)
    listP12.append(x)

else:#---------------------------------------12
        listT1.append(i[1])
        listP1.append(i[3])
        listT2.append(i[5])
        listP2.append(i[7])
        listT3.append(i[9])
        listP3.append(i[11])
        listT4.append(i[13])
```

```
                listP4.append(i[15])
                listT5.append(i[17])
                listP5.append(i[19])
                listT6.append(i[21])
                listP6.append(i[23])
                listT7.append(i[25])
                listP7.append(i[27])
                listT8.append(i[29])
                listP8.append(i[31])
                listT12.append(i[33])
                listP12.append(i[35])


print(data)


#for i in list:
        #data.updateTime = data.updatePriceTimeJson.apply(lambda x: i)
listnum = []
for i in list_len:
        if i == 1:
                listnum.append(0)
        elif i == 5:
                listnum.append(1)
        elif i == 9:
                listnum.append(2)
        elif i == 13:
                listnum.append(3)
        elif i == 17:
                listnum.append(4)
        elif i == 21:
                listnum.append(5)
        elif i == 25:
                listnum.append(6)
        elif i == 29:
                listnum.append(7)
        elif i == 33:
                listnum.append(8)
        elif i == 49:
                listnum.append(12)

data.insert(loc=6, column='updateTime1', value=listT1)
data.insert(loc=7, column='updatePrice1', value=listP1)
data.insert(loc=8, column='updateTime2', value=listT2)
data.insert(loc=9, column='updatePrice2', value=listP2)
data.insert(loc=10, column='updateTime3', value=listT3)
data.insert(loc=11, column='updatePrice3', value=listP3)
data.insert(loc=12, column='updatenum', value=listnum)
# print(data)
# print(Counter(list_len))
# print(Counter(listnum))
data.to_csv("C:\\Users\Lenovo\Desktop\\4_integration.csv", encoding='utf-8')
#print(Counter(data.updateTime))

data1          =          pd.read_csv("C:\\Users\Lenovo\Desktop\\4_integration.csv",
encoding='utf-8')

from pyecharts import options as opts
from pyecharts.charts import Bar
from pyecharts.faker import Faker
dict = dict(Counter(listnum))
print(dict.keys)
listx= [0,1,2,3,4,5,6,7,8,12]
listy = [6763,2126,690,269,97,31,14,5,3,2]
#_________________________________________更新频次数统计
# c = (
#        Bar()
#        .add_xaxis(listx)
#        .add_yaxis("updatenum", listy, color=Faker.rand_color())
#        .set_global_opts(
#               title_opts=opts.TitleOpts(title="更新次数统计"),
#                                               datazoom_opts=[opts.DataZoomOpts(),
opts.DataZoomOpts(type_="inside")],
#        )
#        .render("更新次数统计.html")
# )
sum = 0
for i in data1:
       data1.withdrawPeriod          =          pd.to_datetime(data1.pullDate)          -
pd.to_datetime(data1.pushDate)
print(data1.withdrawPeriod)
list_withdrawPeriod = []
for i in data1.withdrawPeriod:
       list_withdrawPeriod.append(i)
import pandas as pd
data1.withdrawPeriod
=data1.withdrawPeriod.astype('timedelta64[D]').astype(float)
print(data1.withdrawPeriod)
```

```
list_withdrawPeriod = list(data1.withdrawPeriod)
print(list_withdrawPeriod)
print(data1)
data1.insert(loc=14, column='withdrawPeriod', value=list_withdrawPeriod)
#data['L'] = pd.to_datetime(data['LOAD_TIME'])-pd.to_datetime(data['FFP_DATE'])
data1.to_csv("C:\\Users\Lenovo\Desktop\\4_hhh.csv", encoding='utf-8')
data2 = pd.read_csv("C:\\Users\Lenovo\Desktop\\4_hhh.csv", encoding='utf-8')
print("更新次数为 0 的成交周期平均值：",data2[data2.updatenum == 0].withdrawPeriod.mean())
print("更新次数为 1 的成交周期平均值：",data2[data2.updatenum == 1].withdrawPeriod.mean())
print("更新次数为 2 的成交周期平均值：",data2[data2.updatenum == 2].withdrawPeriod.mean())
print("更新次数为 3 的成交周期平均值：",data2[data2.updatenum == 3].withdrawPeriod.mean())
print("更新次数为 4 的成交周期平均值：",data2[data2.updatenum == 4].withdrawPeriod.mean())
print("更新次数为 5 的成交周期平均值：",data2[data2.updatenum == 5].withdrawPeriod.mean())
print("更新次数为 6 的成交周期平均值：",data2[data2.updatenum == 6].withdrawPeriod.mean())
print("更新次数为 7 的成交周期平均值：",data2[data2.updatenum == 7].withdrawPeriod.mean())
print("更新次数为 8 的成交周期平均值：",data2[data2.updatenum == 8].withdrawPeriod.mean())
print("更新次数为 12 的成交周期平均值：",data2[data2.updatenum == 12].withdrawPeriod.mean())
```

2.1.2 数据可视化

```
import pandas as pd
from collections import Counter
data2 = pd.read_csv("C:\\Users\Lenovo\Desktop\\4_hhh.csv", encoding='utf-8')
print(sorted(list(Counter(data2.pushPrice))))
print("价格对应平均成交周期：")
print("小于 5 平均值：",data2[data2.pushPrice < 5].withdrawPeriod.mean())
print("大于 5 平均值：",data2[data2.pushPrice > 5].withdrawPeriod.mean())
print("大于 20 平均值：",data2[data2.pushPrice > 20].withdrawPeriod.mean())
print("大于 50 平均值：",data2[data2.pushPrice > 50].withdrawPeriod.mean())
print("大于 100 平均值：",data2[data2.pushPrice > 100].withdrawPeriod.mean())
print("大于 500 平均值：",data2[data2.pushPrice > 500].withdrawPeriod.mean())
print(dict(Counter(data2.pushPrice)))
```

```
a = dict(Counter(data2.pushPrice))
import pyecharts.options as opts

x_data = list(a.keys())
y_data = list(a.values())
from pyecharts.charts import Scatter

(
        Scatter(init_opts=opts.InitOpts(width="1600px", height="1000px"))
                .add_xaxis(xaxis_data=x_data)
                .add_yaxis(
                series_name="",
                y_axis=y_data,
                symbol_size=20,
                label_opts=opts.LabelOpts(is_show=False),
        )
                .set_series_opts()
                .set_global_opts(

                xaxis_opts=opts.AxisOpts(
                        type_="value", splitline_opts=opts.SplitLineOpts(is_show=True)
                ),
                yaxis_opts=opts.AxisOpts(
                        type_="value",
                        axistick_opts=opts.AxisTickOpts(is_show=True),
                        splitline_opts=opts.SplitLineOpts(is_show=True),

                ),
                tooltip_opts=opts.TooltipOpts(is_show=False),
        )
                .render("价格—价格出现次数次数散点图.html")
)


import pandas as pd
from collections import Counter
data2 = pd.read_csv("C:\\Users\Lenovo\Desktop\\4_hhh.csv", encoding='utf-8')
x1 = data2[data2.withdrawDate == "0"].updatenum
print("未成交车辆的更新频次统计：",data2[data2.withdrawDate ==
"0"].updatenum.count())
print(x1)
print("未成交车辆的更新频次统计的 Counter 的结果：",Counter(x1))
#data[data.isevening==1].Dept.count()
x2 = data2[data2.withdrawDate != "0"].updatenum
```

```
print(" 成 交 车 辆 的 更 新 频 次 统 计 ： ",data2[data2.withdrawDate !=
"0"].updatenum.count())
print(x2)
print("成交车辆的更新频次统计的 Counter 的结果：",Counter(x2))
#data[data.isevening==1].Dept.count()
from pyecharts import options as opts
from pyecharts.charts import Bar
from pyecharts.commons.utils import JsCode
from pyecharts.globals import ThemeType

list2 = [
      {"value": 5304, "percent": 5304 / (5304 + 1459)},
      {"value": 1763, "percent": 1763 / (1763 + 363)},
      {"value": 586, "percent": 586 / (586 + 104)},
      {"value": 220, "percent": 220 / (220 + 49)},
      {"value": 80, "percent": 80 / (80 + 17)},
      {"value": 25, "percent": 25 / (25 + 6)},
{"value": 13, "percent": 13 / (13 + 1)},
{"value": 4, "percent": 4 / (4 + 1)},
{"value": 3, "percent": 3 / (3 + 0)},
{"value": 2, "percent": 2 / (2 + 0)},
]

list3 = [
      {"value": 1459, "percent": 1459 / (1459 + 5304)},
      {"value": 363, "percent": 363 / (363 + 1763)},
      {"value": 104, "percent": 104 / (104+ 586)},
      {"value": 49, "percent": 49/ (49 + 220)},
      {"value": 17, "percent": 17 / (17 + 80)},
{"value": 6, "percent": 6 / (6 + 25)},
{"value": 1, "percent": 1 / (1 + 13)},
{"value": 1, "percent": 1 / (1 + 4)},
{"value": 0, "percent": 0 / (0 + 3)},
{"value": 0, "percent": 0 / (0 + 2)},
]

c = (
      Bar(init_opts=opts.InitOpts(theme=ThemeType.LIGHT))
      .add_xaxis([0,1, 2, 3, 4, 5,6,7,8,12])
      .add_yaxis("成交车辆", list2, stack="stack1", category_gap="10%")
      .add_yaxis("未成交车辆", list3, stack="stack2", category_gap="50%")
      .set_series_opts(
            label_opts=opts.LabelOpts(
                  position="right",
                formatter=JsCode(
                    "function(x){return  Number(x.data.percent  *  100).toFixed()  +
'%';}"
                ),
            )
        )
        .render("成交车辆与未成交车辆更新频次占比图.html")
)

from pyecharts import options as opts
from pyecharts.charts import Bar
from pyecharts.faker import Faker

c = (
    Bar()
    .add_xaxis([0,1, 2, 3, 4, 5,6,7,8,12])
        .add_yaxis("成交", [5304, 1763, 586, 220, 80, 25, 13, 4, 3, 2])
        .add_yaxis("未成交", [1459, 363, 104, 49, 17, 6, 1, 1, 0, 0])
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Bar-显示  ToolBox"),
        toolbox_opts=opts.ToolboxOpts(),
        legend_opts=opts.LegendOpts(is_show=False),
    )
    .render("成交车辆与未成交车辆更新频次对比图.html")
)

import pandas as pd
from collections import Counter
data2 = pd.read_csv("C:\\Users\Lenovo\Desktop\\4_hhh.csv", encoding='utf-8')
print(" 未 成 交 车 辆 更 新 次 数 平 均 值 ： ",data2[data2.withdrawDate  ==
"0"].updatenum.mean())
print(" 成 交 车 辆 更 新 次 数 平 均 值 ： ",data2[data2.withdrawDate  !=
"0"].updatenum.mean())
import pyecharts.options as opts
x_data = data2.updatenum.tolist()
y_data = data2.withdrawPeriod.tolist()
from pyecharts.charts import Scatter

(
    Scatter(init_opts=opts.InitOpts(width="1600px", height="1000px"))
    .add_xaxis(xaxis_data=x_data)
    .add_yaxis(
        series_name="",
        y_axis=y_data,
            symbol_size=20,
            label_opts=opts.LabelOpts(is_show=False),
        )
        .set_series_opts()
        .set_global_opts(

            xaxis_opts=opts.AxisOpts(
                type_="value", splitline_opts=opts.SplitLineOpts(is_show=True)
            ),
            yaxis_opts=opts.AxisOpts(
                type_="value",
                axistick_opts=opts.AxisTickOpts(is_show=True),
                splitline_opts=opts.SplitLineOpts(is_show=True),

            ),
            tooltip_opts=opts.TooltipOpts(is_show=False),
        )
        .render("更新次数—成交周期散点图.html")
)
```

2.2 综合附件一与附件四

2.2.1 数据分析

```
# 数据处理
import numpy as np
import pandas as pd
import datetime

# 读取附件四
df = pd.read_table('./data/附件 4：门店交易训练数据.txt', sep='\t', header=None,
encoding='gbk')

# 修改中文标签
columns = ['车辆 id', '上架时间', '上架价格', '{价格调整时间：调整后价格}', '下架
时间(成交车辆下架时间和成交时间相同)', '成交时间']
df.columns = columns

# 保存文件附件 4-1（附件四含中文标签）
df.to_csv('./data/附件 4-1.csv', index = False)
```

```
# 读取附件 1
df1 = pd.read_table('./data/附件 1：估价训练数据.txt', sep='\t', header=None, 
encoding='gbk')
# 修改中文标签
columns1 = ['车辆 id', '展销时间', '品牌 id ', '车系 id ', '车型 id ', '里程', '车辆颜
色', '车辆所在城市 id', '国标码', '过户次数', '载客人数', '注册日期', '上牌日期', '国
别', '厂商类型', '年款', '排量', '变速箱', '燃油类型', '新车价', '匿名特征', '匿名特征', 
'匿名特征', '匿名特征', '匿名特征', '匿名特征', '匿名特征', '匿名特征', '匿名特征', '
匿名特征', '匿名特征', '匿名特征', '匿名特征', '匿名特征', '匿名特征', '二手车交易
价格（预测目标）']
df1.columns = columns1
# 保存
df1.to_csv('./data/附件 1-1.csv', index = False)

# 合并附件一与附件四
df2 = pd.merge(df1, df, how = 'inner', on = ['车辆 id'])     # 按照车辆 id 合并
df2.to_csv('./data/附件 4-2.csv', index = False)

# 整理销售周期（天数）
df2['上架时间'] = pd.to_datetime(df['上架时间'])
df2['下架时间(成交车辆下架时间和成交时间相同)'] = pd.to_datetime(df2['下架
时间(成交车辆下架时间和成交时间相同)'])
df2['成交周期(天)'] = df2['下架时间(成交车辆下架时间和成交时间相同)'] - df2['
上架时间']
df2['成交周期(天)'] = df2['成交周期(天)'].apply(lambda x: x.days)
df2.to_csv('./data/附件 4-3（含销售周期天数）.csv', index = False)

# 划分销售周期区间
cycle_bins = [0, 1, 2, 3, 4, 5, 8, 18, 51, 92, 278]
cycle_labels = ['0', '1', '2', '3', '4', '5-7', '8-17', '18-50', '51-91', '92-277']
a, b = pd.cut(x = df2['成交周期(天)'], bins = cycle_bins, right = False, labels = 
cycle_labels, retbins = True)
df2['周期区间'], b = pd.cut(x = df2['成交周期(天)'], bins = cycle_bins, right = False, 
retbins = True)
df2['周期标签'], b = pd.cut(x = df2['成交周期(天)'], bins = cycle_bins, right = False, 
labels = cycle_labels, retbins = True)

df2.to_csv('./data/附件 4-4（含周期区间）.csv', index = False)
```

2.2.2 销售周期天数分析

```
import pandas as pd
import numpy as np
from collections import Counter

from pyecharts import options as opts
from pyecharts.charts import Scatter
from pyecharts.charts import Line

def counter(list):
    from collections import Counter
    # 对列表内的元素进行排序，计数，以列表的形式返回
    list_counter = Counter(list)
    list_sorted = sorted(list_counter.items(), key = lambda x:x[0], reverse = False)
    # lambda x:x[1]按值排序，reverse = True 倒序 即大值在前。
    list_dict = dict(list_sorted)
    return list_dict

# 对销售周期天数的分析
df = pd.read_csv('./data/附件 4-3（含销售周期天数）.csv')
transaction = list(df['成交周期(天)'])
# print(max(transaction))
# print(min(transaction))
cycle = list(range(0, 278))
transaction_counter = Counter(transaction)
transaction_sorted = sorted(transaction_counter.items(), key = lambda x: x[0],
reverse = False)
transaction_dict = dict(transaction_sorted)

transaction_sorted1 = sorted(transaction_counter.items(), key = lambda x: x[1],
reverse = False)
transaction_dict1 = dict(transaction_sorted1)

# 绘制销售周期的散点图
c = (
    Scatter()
    .add_xaxis(list(transaction_dict))
    .add_yaxis(
        "次",
        list(transaction_dict.values()),
    )
    .set_global_opts(
            title_opts = opts.TitleOpts(title = '销售周期散点图')
        )
        .render('./charts/销售周期散点图.html')
)

# 比较品牌 id 对销售周期的影响
brandid_dict = counter(list(df['品牌  id ']))
# 绘制折线图显示
c = (
        Line()
        .add_xaxis(xaxis_data=list(brandid_dict))
        .add_yaxis("数量", y_axis=list(brandid_dict.values()))
        .render('./charts/品牌 id 对销售周期的影响折线图.html')
)

brandid_dict1 = sorted(brandid_dict.items(), key = lambda x:x[1], reverse = True)

# 整体销量前十的品牌
brandid_top10 = dict(brandid_dict1[:10])
brand_top10 = list(brandid_top10)

brandeal = df.groupby('品牌  id ')['成交周期(天)'].apply(list).to_dict()
# 每个周期出现多少品牌
dealbrand = df.groupby('成交周期(天)')['品牌  id '].apply(list).to_dict()
brand_0 = brandeal[brand_top10[0]]
brand_1 = brandeal[brand_top10[1]]
brand_2 = brandeal[brand_top10[2]]
brand_3 = brandeal[brand_top10[3]]
brand_4 = brandeal[brand_top10[4]]
brand_5 = brandeal[brand_top10[5]]
brand_6 = brandeal[brand_top10[6]]
brand_7 = brandeal[brand_top10[7]]
brand_8 = brandeal[brand_top10[8]]
brand_9 = brandeal[brand_top10[9]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", brand_0)
        .add_yaxis("top_2", brand_1)
        .add_yaxis("top_3", brand_2)
        .add_yaxis("top_4", brand_3)
        .add_yaxis("top_5", brand_4)
        .add_yaxis("top_6", brand_5)
        .add_yaxis("top_7", brand_6)
```

```
    .add_yaxis("top_8", brand_7)
    .add_yaxis("top_9", brand_8)
    .add_yaxis("top_10", brand_9)
#     .set_global_opts(
#         title_opts = opts.TitleOpts(title = "品牌前 10 的销售周期")
#     )
    .render('./charts/品牌前 10 的销售周期散点图.html')
)

# 销量最少的十个
brandid_dreciprocal10 = dict(brandid_dict1[-10:])
print(brandid_dreciprocal10)
# len(brandid_reciprocal10)
brand_reciprocal10 = list(brandid_dreciprocal10)
print(brand_reciprocal10)
brand_d0 = brandeal[brand_reciprocal10[0]]
brand_d1 = brandeal[brand_reciprocal10[1]]
brand_d2 = brandeal[brand_reciprocal10[2]]
brand_d3 = brandeal[brand_reciprocal10[3]]
brand_d4 = brandeal[brand_reciprocal10[4]]
brand_d5 = brandeal[brand_reciprocal10[5]]
brand_d6 = brandeal[brand_reciprocal10[6]]
brand_d7 = brandeal[brand_reciprocal10[7]]
brand_d8 = brandeal[brand_reciprocal10[8]]
brand_d9 = brandeal[brand_reciprocal10[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("reci_1", brand_d0)
    .add_yaxis("reci_2", brand_d1)
    .add_yaxis("reci_3", brand_d2)
    .add_yaxis("reci_4", brand_d3)
    .add_yaxis("reci_5", brand_d4)
    .add_yaxis("reci_6", brand_d5)
    .add_yaxis("reci_7", brand_d6)
    .add_yaxis("reci_8", brand_d7)
    .add_yaxis("reci_9", brand_d8)
    .add_yaxis("reci_10", brand_d9)
#     .set_global_opts(
#         title_opts = opts.TitleOpts(title = "品牌后 10 的销售周期")
#     )
    .render('./charts/品牌后 10 的销售周期散点图.html')
)
c.render_notebook()
```

```
# 车系 id
chexiid_dict = counter(list(df['车系  id ']))
print("len: ", len(chexiid_dict))
print("max: ", max(chexiid_dict))
print("min: ", min(chexiid_dict))
# 绘制折线图显示
c = (
        Line()
        .add_xaxis(xaxis_data=list(chexiid_dict))
        .add_yaxis("数量", y_axis=list(chexiid_dict.values()))
        .render('./charts/车系 id 销量折线图.html')
)
# 按销量排序
chexiid_dict1 = sorted(chexiid_dict.items(), key = lambda x:x[1], reverse = True)
# 整理车系 id 与成交周期
chexideal = df.groupby('车系  id ')['成交周期(天)'].apply(list).to_dict()
# 车系销量前十
chexiid_top10 = dict(chexiid_dict1[:10])
print(chexiid_top10)
chexi_top10 = list(chexiid_top10)
print(chexi_top10)
# 前 10
chexi_0 = chexideal[chexi_top10[0]]
chexi_1 = chexideal[chexi_top10[1]]
chexi_2 = chexideal[chexi_top10[2]]
chexi_3 = chexideal[chexi_top10[3]]
chexi_4 = chexideal[chexi_top10[4]]
chexi_5 = chexideal[chexi_top10[5]]
chexi_6 = chexideal[chexi_top10[6]]
chexi_7 = chexideal[chexi_top10[7]]
chexi_8 = chexideal[chexi_top10[8]]
chexi_9 = chexideal[chexi_top10[9]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("1", chexi_0)
        .add_yaxis("2", chexi_1)
        .add_yaxis("3", chexi_2)
        .add_yaxis("4", chexi_3)
        .add_yaxis("5", chexi_4)
        .add_yaxis("6", chexi_5)
        .add_yaxis("7", chexi_6)
        .add_yaxis("8", chexi_7)
```

```
    .add_yaxis("9", chexi_8)
    .add_yaxis("10", chexi_9)
#       .set_global_opts(
#               title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#       )
    .render('./charts/车系 id 前 10 的销售周期散点图.html')
)

# 车系销量后十
chexiid_dreciprocal10 = dict(chexiid_dict1[-10:])
print(chexiid_dreciprocal10)
chexi_reciprocal10 = list(chexiid_dreciprocal10)
print(chexi_reciprocal10)
# 后 10
chexiid_dreciprocal10 = dict(chexiid_dict1[-10:])
print(chexiid_dreciprocal10)
# len(chexiid_reciprocal10)
chexi_reciprocal10 = list(chexiid_dreciprocal10)
print(chexi_reciprocal10)
chexi_d0 = chexideal[chexi_reciprocal10[0]]
chexi_d1 = chexideal[chexi_reciprocal10[1]]
chexi_d2 = chexideal[chexi_reciprocal10[2]]
chexi_d3 = chexideal[chexi_reciprocal10[3]]
chexi_d4 = chexideal[chexi_reciprocal10[4]]
chexi_d5 = chexideal[chexi_reciprocal10[5]]
chexi_d6 = chexideal[chexi_reciprocal10[6]]
chexi_d7 = chexideal[chexi_reciprocal10[7]]
chexi_d8 = chexideal[chexi_reciprocal10[8]]
chexi_d9 = chexideal[chexi_reciprocal10[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("reci_1", chexi_d0)
    .add_yaxis("reci_2", chexi_d1)
    .add_yaxis("reci_3", chexi_d2)
    .add_yaxis("reci_4", chexi_d3)
    .add_yaxis("reci_5", chexi_d4)
    .add_yaxis("reci_6", chexi_d5)
    .add_yaxis("reci_7", chexi_d6)
    .add_yaxis("reci_8", chexi_d7)
    .add_yaxis("reci_9", chexi_d8)
    .add_yaxis("reci_10", chexi_d9)
#       .set_global_opts(
#               title_opts = opts.TitleOpts(title = "后 10 的销售周期")
#         )
        .render('./charts/车系 id 后 10 的销售周期散点图.html')
)

# 车型 id
chexingid_dict = counter(list(df['车型  id ']))
print("len: ", len(chexingid_dict))
print("max: ", max(chexingid_dict))
print("min: ", min(chexingid_dict))
# 绘制折线图显示
c = (
        Line()
        .add_xaxis(xaxis_data=list(chexingid_dict))
        .add_yaxis("数量", y_axis=list(chexingid_dict.values()))
        .render('./charts/车型 id 销量折线图.html')
)
# 按销量排序
chexingid_dict1 = sorted(chexingid_dict.items(), key = lambda x:x[1], reverse =
True)
# 整理车系 id 与成交周期
chexingdeal = df.groupby('车型  id ')['成交周期(天)'].apply(list).to_dict()
# 前 10
chexingid_top10 = dict(chexingid_dict1[:10])
print(chexingid_top10)
chexing_top10 = list(chexingid_top10)
print(chexing_top10)
chexing_0 = chexingdeal[chexing_top10[0]]
chexing_1 = chexingdeal[chexing_top10[1]]
chexing_2 = chexingdeal[chexing_top10[2]]
chexing_3 = chexingdeal[chexing_top10[3]]
chexing_4 = chexingdeal[chexing_top10[4]]
chexing_5 = chexingdeal[chexing_top10[5]]
chexing_6 = chexingdeal[chexing_top10[6]]
chexing_7 = chexingdeal[chexing_top10[7]]
chexing_8 = chexingdeal[chexing_top10[8]]
chexing_9 = chexingdeal[chexing_top10[9]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", chexing_0)
        .add_yaxis("top_2", chexing_1)
        .add_yaxis("top_3", chexing_2)
        .add_yaxis("top_4", chexing_3)
        .add_yaxis("top_5", chexing_4)
        .add_yaxis("top_6", chexing_5)
        .add_yaxis("top_7", chexing_6)
        .add_yaxis("top_8", chexing_7)
        .add_yaxis("top_9", chexing_8)
        .add_yaxis("top_10", chexing_9)
#       .set_global_opts(
#               title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#       )
        .render('./charts/车系 id 销量前 10 销售周期散点图.html')
)

# 后 10
chexingid_dreciprocal10 = dict(chexingid_dict1[-10:])
print(chexingid_dreciprocal10)
# len(chexingid_reciprocal10)
chexing_reciprocal10 = list(chexingid_dreciprocal10)
print(chexing_reciprocal10)
chexing_d0 = chexingdeal[chexing_reciprocal10[0]]
chexing_d1 = chexingdeal[chexing_reciprocal10[1]]
chexing_d2 = chexingdeal[chexing_reciprocal10[2]]
chexing_d3 = chexingdeal[chexing_reciprocal10[3]]
chexing_d4 = chexingdeal[chexing_reciprocal10[4]]
chexing_d5 = chexingdeal[chexing_reciprocal10[5]]
chexing_d6 = chexingdeal[chexing_reciprocal10[6]]
chexing_d7 = chexingdeal[chexing_reciprocal10[7]]
chexing_d8 = chexingdeal[chexing_reciprocal10[8]]
chexing_d9 = chexingdeal[chexing_reciprocal10[9]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("reci_1", chexing_d0)
        .add_yaxis("reci_2", chexing_d1)
        .add_yaxis("reci_3", chexing_d2)
        .add_yaxis("reci_4", chexing_d3)
        .add_yaxis("reci_5", chexing_d4)
        .add_yaxis("reci_6", chexing_d5)
        .add_yaxis("reci_7", chexing_d6)
        .add_yaxis("reci_8", chexing_d7)
        .add_yaxis("reci_9", chexing_d8)
        .add_yaxis("reci_10", chexing_d9)
#       .set_global_opts(
#               title_opts = opts.TitleOpts(title = "后 10 的销售周期")
#       )
        .render('./charts/车系 id 销量后 10 销售周期散点图.html')
```

```
)

# 里程
# 排序，按里程
licheng_dict = counter(list(df['里程']))
print("len: ", len(licheng_dict))
print( "max: " , max(list(df['里程'])))
print( "min: " , min(list(df['里程'])))
# 绘制折线图显示
c = (
    Line()
    .add_xaxis(xaxis_data = list(licheng_dict))
    .add_yaxis("数量", y_axis = list(licheng_dict.values()))
    .render('./charts/里程对销售周期的影响折线图.html')
)
# 按销量排序
licheng_dict1 = sorted(licheng_dict.items(), key = lambda x:x[1], reverse = True)
# 整理里程与成交周期
lichengdeal = df.groupby('里程')['成交周期(天)'].apply(list).to_dict()
# 前 10
dlicheng_top10 = dict(licheng_dict1[:10])
print(dlicheng_top10)
licheng_top10 = list(dlicheng_top10)
print(licheng_top10)
licheng_0 = lichengdeal[licheng_top10[0]]
licheng_1 = lichengdeal[licheng_top10[1]]
licheng_2 = lichengdeal[licheng_top10[2]]
licheng_3 = lichengdeal[licheng_top10[3]]
licheng_4 = lichengdeal[licheng_top10[4]]
licheng_5 = lichengdeal[licheng_top10[5]]
licheng_6 = lichengdeal[licheng_top10[6]]
licheng_7 = lichengdeal[licheng_top10[7]]
licheng_8 = lichengdeal[licheng_top10[8]]
licheng_9 = lichengdeal[licheng_top10[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("top_1", licheng_0)
    .add_yaxis("top_2", licheng_1)
    .add_yaxis("top_3", licheng_2)
    .add_yaxis("top_4", licheng_3)
    .add_yaxis("top_5", licheng_4)
    .add_yaxis("top_6", licheng_5)
    .add_yaxis("top_7", licheng_6)
    .add_yaxis("top_8", licheng_7)
    .add_yaxis("top_9", licheng_8)
    .add_yaxis("top_10", licheng_9)
#       .set_global_opts(
#           title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#       )
    .render('./charts/里程销量前 10 销售周期散点图.html')
)
# 后 10
dlicheng_dreciprocal10 = dict(licheng_dict1[-10:])
print(dlicheng_dreciprocal10)
# len(dlicheng_reciprocal10)
licheng_reciprocal10 = list(dlicheng_dreciprocal10)
print(licheng_reciprocal10)
licheng_d0 = lichengdeal[licheng_reciprocal10[0]]
licheng_d1 = lichengdeal[licheng_reciprocal10[1]]
licheng_d2 = lichengdeal[licheng_reciprocal10[2]]
licheng_d3 = lichengdeal[licheng_reciprocal10[3]]
licheng_d4 = lichengdeal[licheng_reciprocal10[4]]
licheng_d5 = lichengdeal[licheng_reciprocal10[5]]
licheng_d6 = lichengdeal[licheng_reciprocal10[6]]
licheng_d7 = lichengdeal[licheng_reciprocal10[7]]
licheng_d8 = lichengdeal[licheng_reciprocal10[8]]
licheng_d9 = lichengdeal[licheng_reciprocal10[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("reci_1", licheng_d0)
    .add_yaxis("reci_2", licheng_d1)
    .add_yaxis("reci_3", licheng_d2)
    .add_yaxis("reci_4", licheng_d3)
    .add_yaxis("reci_5", licheng_d4)
    .add_yaxis("reci_6", licheng_d5)
    .add_yaxis("reci_7", licheng_d6)
    .add_yaxis("reci_8", licheng_d7)
    .add_yaxis("reci_9", licheng_d8)
    .add_yaxis("reci_10", licheng_d9)
#       .set_global_opts(
#           title_opts = opts.TitleOpts(title = "后 10 的销售周期")
#       )
    .render('./charts/里程销量后 10 销售周期散点图.html')
)

# 车辆颜色
# 排序，按车辆颜色
color_dict = counter(list(df['车辆颜色']))
# 按销量排序
color_dict1 = sorted(color_dict.items(), key = lambda x:x[1], reverse = True)
# 整理车辆颜色与成交周期
colordeal = df.groupby('车辆颜色')['成交周期(天)'].apply(list).to_dict()

dcolor_top10 = dict(color_dict1[:10])
print(dcolor_top10)
color_top10 = list(dcolor_top10)
print(color_top10)
color_0 = colordeal[color_top10[0]]
color_1 = colordeal[color_top10[1]]
color_2 = colordeal[color_top10[2]]
color_3 = colordeal[color_top10[3]]
color_4 = colordeal[color_top10[4]]
color_5 = colordeal[color_top10[5]]
color_6 = colordeal[color_top10[6]]
color_7 = colordeal[color_top10[7]]
color_8 = colordeal[color_top10[8]]
color_9 = colordeal[color_top10[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("top_1", color_0)
    .add_yaxis("top_2", color_1)
    .add_yaxis("top_3", color_2)
    .add_yaxis("top_4", color_3)
    .add_yaxis("top_5", color_4)
    .add_yaxis("top_6", color_5)
    .add_yaxis("top_7", color_6)
    .add_yaxis("top_8", color_7)
    .add_yaxis("top_9", color_8)
    .add_yaxis("top_10", color_9)
#       .set_global_opts(
#           title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#       )
    .render('./charts/颜色对交易周期影响分布图.html')
)

# 车辆所在城市 id
# 排序，按车辆所在城市  id
city_dict = counter(list(df['车辆所在城市  id']))
# 绘制折线图显示
```

```
c = (
        Line()
        .add_xaxis(xaxis_data=list(city_dict))
        .add_yaxis("数量", y_axis=list(city_dict.values()))
        .render('./charts/车辆所在城市对销售周期的影响折线图.html')
)
# 按销量排序
city_dict1 = sorted(city_dict.items(), key = lambda x:x[1], reverse = True)
# 整理车辆所在城市 id 与成交周期
citydeal = df.groupby('车辆所在城市 id')['成交周期(天)'].apply(list).to_dict()
# 前 10
dcity_top10 = dict(city_dict1[:10])
print(dcity_top10)
city_top10 = list(dcity_top10)
print(city_top10)
city_0 = citydeal[city_top10[0]]
city_1 = citydeal[city_top10[1]]
city_2 = citydeal[city_top10[2]]
city_3 = citydeal[city_top10[3]]
city_4 = citydeal[city_top10[4]]
city_5 = citydeal[city_top10[5]]
city_6 = citydeal[city_top10[6]]
city_7 = citydeal[city_top10[7]]
city_8 = citydeal[city_top10[8]]
city_9 = citydeal[city_top10[9]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", city_0)
        .add_yaxis("top_2", city_1)
        .add_yaxis("top_3", city_2)
        .add_yaxis("top_4", city_3)
        .add_yaxis("top_5", city_4)
        .add_yaxis("top_6", city_5)
        .add_yaxis("top_7", city_6)
        .add_yaxis("top_8", city_7)
        .add_yaxis("top_9", city_8)
        .add_yaxis("top_10", city_9)
#           .set_global_opts(
#                   title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#           )
        .render('./charts/车辆所在城市销量前 10 销售周期散点图.html')
)
```

```
# 后 10
dcity_dreciprocal10 = dict(city_dict1[-10:])
print(dcity_dreciprocal10)
# len(dcity_reciprocal10)
city_reciprocal10 = list(dcity_dreciprocal10)
print(city_reciprocal10)
city_d0 = citydeal[city_reciprocal10[0]]
city_d1 = citydeal[city_reciprocal10[1]]
city_d2 = citydeal[city_reciprocal10[2]]
city_d3 = citydeal[city_reciprocal10[3]]
city_d4 = citydeal[city_reciprocal10[4]]
city_d5 = citydeal[city_reciprocal10[5]]
city_d6 = citydeal[city_reciprocal10[6]]
city_d7 = citydeal[city_reciprocal10[7]]
city_d8 = citydeal[city_reciprocal10[8]]
city_d9 = citydeal[city_reciprocal10[9]]
c = (
	Scatter()
	.add_xaxis(cycle)
	.add_yaxis("reci_1", city_d0)
	.add_yaxis("reci_2", city_d1)
	.add_yaxis("reci_3", city_d2)
	.add_yaxis("reci_4", city_d3)
	.add_yaxis("reci_5", city_d4)
	.add_yaxis("reci_6", city_d5)
	.add_yaxis("reci_7", city_d6)
	.add_yaxis("reci_8", city_d7)
	.add_yaxis("reci_9", city_d8)
	.add_yaxis("reci_10", city_d9)
#	  .set_global_opts(
#		title_opts = opts.TitleOpts(title = "后 10 的销售周期")
#	  )
	.render('./charts/车辆所在城市销量后 10 销售周期散点图.html')
)

# 过户次数
# 排序，按过户次数
guohucount_dict = counter(list(df['过户次数']))
# 绘制折线图显示
c = (
	Line()
	.add_xaxis(xaxis_data = list(guohucount_dict))
	.add_yaxis("数量", y_axis = list(guohucount_dict.values()))
	.render('./charts/过户次数对销售周期的影响折线图.html')
)
# 按销量排序
guohucount_dict1 = sorted(guohucount_dict.items(), key = lambda x:x[1], reverse
= True)
# 整理过户次数与成交周期
guohucountdeal = df.groupby('过户次数')['成交周期(天)'].apply(list).to_dict()
# 前 10
dguohucount_top10 = dict(guohucount_dict1[:10])
print(dguohucount_top10)
guohucount_top10 = list(dguohucount_top10)
print(guohucount_top10)
guohucount_0 = guohucountdeal[guohucount_top10[0]]
guohucount_1 = guohucountdeal[guohucount_top10[1]]
guohucount_2 = guohucountdeal[guohucount_top10[2]]
guohucount_3 = guohucountdeal[guohucount_top10[3]]
guohucount_4 = guohucountdeal[guohucount_top10[4]]
guohucount_5 = guohucountdeal[guohucount_top10[5]]
guohucount_6 = guohucountdeal[guohucount_top10[6]]
guohucount_7 = guohucountdeal[guohucount_top10[7]]
guohucount_8 = guohucountdeal[guohucount_top10[8]]
guohucount_9 = guohucountdeal[guohucount_top10[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("top_1", guohucount_0)
    .add_yaxis("top_2", guohucount_1)
    .add_yaxis("top_3", guohucount_2)
    .add_yaxis("top_4", guohucount_3)
    .add_yaxis("top_5", guohucount_4)
    .add_yaxis("top_6", guohucount_5)
    .add_yaxis("top_7", guohucount_6)
    .add_yaxis("top_8", guohucount_7)
    .add_yaxis("top_9", guohucount_8)
    .add_yaxis("top_10", guohucount_9)
#      .set_global_opts(
#          title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#      )
    .render('./charts/ 过户次数对销售周期散点图.html')
)

# 载客人数
# 排序，按载客人数
zaikerenshu_dict = counter(list(df['载客人数']))
# 绘制折线图显示
```

```
c = (
        Line()
        .add_xaxis(list(zaikerenshu_dict))
        .add_yaxis("数量", list(zaikerenshu_dict.values()))
        .render('./charts/载客人数对销售周期的影响折线图.html')
)
# 按销量排序
zaikerenshu_dict1 = sorted(zaikerenshu_dict.items(), key = lambda x:x[1], reverse
= True)
# 整理载客人数与成交周期
zaikerenshudeal = df.groupby('载客人数')['成交周期(天)'].apply(list).to_dict()
# 前 10
zaikerenshu_top10 = list(zaikerenshudeal)
print(zaikerenshu_top10)
zaikerenshu_0 = zaikerenshudeal[zaikerenshu_top10[0]]
zaikerenshu_1 = zaikerenshudeal[zaikerenshu_top10[1]]
zaikerenshu_2 = zaikerenshudeal[zaikerenshu_top10[2]]
zaikerenshu_3 = zaikerenshudeal[zaikerenshu_top10[3]]
zaikerenshu_4 = zaikerenshudeal[zaikerenshu_top10[4]]
zaikerenshu_5 = zaikerenshudeal[zaikerenshu_top10[5]]
zaikerenshu_6 = zaikerenshudeal[zaikerenshu_top10[6]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", zaikerenshu_0)
        .add_yaxis("top_2", zaikerenshu_1)
        .add_yaxis("top_3", zaikerenshu_2)
        .add_yaxis("top_4", zaikerenshu_3)
        .add_yaxis("top_5", zaikerenshu_4)
        .add_yaxis("top_6", zaikerenshu_5)
        .add_yaxis("top_7", zaikerenshu_6)
#           .set_global_opts(
#                   title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#           )
        .render('./charts/载客人数对销售周期影响散点图.html')
)

# 排量
# 排序，按排量
pailiang_dict = counter(list(df['排量']))
# 绘制折线图显示
c = (
        Line()
        .add_xaxis(xaxis_data = list(pailiang_dict))
        .add_yaxis("数量", y_axis = list(pailiang_dict.values()))
        .render('./charts/排量对销售周期的影响折线图.html')
)
# 按销量排序
pailiang_dict1 = sorted(pailiang_dict.items(), key = lambda x:x[1], reverse = True)
# 整理排量与成交周期
pailiangdeal = df.groupby('排量')['成交周期(天)'].apply(list).to_dict()
# 前 10
dpailiang_top10 = dict(pailiang_dict1[:10])
print(dpailiang_top10)
pailiang_top10 = list(dpailiang_top10)
print(pailiang_top10)
pailiang_0 = pailiangdeal[pailiang_top10[0]]
pailiang_1 = pailiangdeal[pailiang_top10[1]]
pailiang_2 = pailiangdeal[pailiang_top10[2]]
pailiang_3 = pailiangdeal[pailiang_top10[3]]
pailiang_4 = pailiangdeal[pailiang_top10[4]]
pailiang_5 = pailiangdeal[pailiang_top10[5]]
pailiang_6 = pailiangdeal[pailiang_top10[6]]
pailiang_7 = pailiangdeal[pailiang_top10[7]]
pailiang_8 = pailiangdeal[pailiang_top10[8]]
pailiang_9 = pailiangdeal[pailiang_top10[9]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", pailiang_0)
        .add_yaxis("top_2", pailiang_1)
        .add_yaxis("top_3", pailiang_2)
        .add_yaxis("top_4", pailiang_3)
        .add_yaxis("top_5", pailiang_4)
        .add_yaxis("top_6", pailiang_5)
        .add_yaxis("top_7", pailiang_6)
        .add_yaxis("top_8", pailiang_7)
        .add_yaxis("top_9", pailiang_8)
        .add_yaxis("top_10", pailiang_9)
#         .set_global_opts(
#                 title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#         )
        .render('./charts/排量销量前 10 销售周期散点图.html')
)
# 后 10
dpailiang_dreciprocal10 = dict(pailiang_dict1[-10:])
print(dpailiang_dreciprocal10)
# len(dpailiang_reciprocal10)
```

```
pailiang_reciprocal10 = list(dpailiang_dreciprocal10)
print(pailiang_reciprocal10)
pailiang_d0 = pailiangdeal[pailiang_reciprocal10[0]]
pailiang_d1 = pailiangdeal[pailiang_reciprocal10[1]]
pailiang_d2 = pailiangdeal[pailiang_reciprocal10[2]]
pailiang_d3 = pailiangdeal[pailiang_reciprocal10[3]]
pailiang_d4 = pailiangdeal[pailiang_reciprocal10[4]]
pailiang_d5 = pailiangdeal[pailiang_reciprocal10[5]]
pailiang_d6 = pailiangdeal[pailiang_reciprocal10[6]]
pailiang_d7 = pailiangdeal[pailiang_reciprocal10[7]]
pailiang_d8 = pailiangdeal[pailiang_reciprocal10[8]]
pailiang_d9 = pailiangdeal[pailiang_reciprocal10[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("reci_1", pailiang_d0)
    .add_yaxis("reci_2", pailiang_d1)
    .add_yaxis("reci_3", pailiang_d2)
    .add_yaxis("reci_4", pailiang_d3)
    .add_yaxis("reci_5", pailiang_d4)
    .add_yaxis("reci_6", pailiang_d5)
    .add_yaxis("reci_7", pailiang_d6)
    .add_yaxis("reci_8", pailiang_d7)
    .add_yaxis("reci_9", pailiang_d8)
    .add_yaxis("reci_10", pailiang_d9)
#     .set_global_opts(
#         title_opts = opts.TitleOpts(title = "后 10 的销售周期")
#     )
    .render('./charts/排量销量后 10 销售周期散点图.html')
)

# 变速箱
# 排序，按变速箱
biansuxiang_dict = counter(list(df['变速箱']))
# 绘制折线图显示
c = (
    Line()
    .add_xaxis(xaxis_data = list(biansuxiang_dict))
    .add_yaxis("数量", y_axis = list(biansuxiang_dict.values()))
    .render('./charts/变速箱对销售周期的影响折线图.html')
)
# 按销量排序
biansuxiang_dict1 = sorted(biansuxiang_dict.items(), key = lambda x:x[1], reverse
= True)
```

```
# 整理变速箱与成交周期
biansuxiangdeal = df.groupby('变速箱')['成交周期(天)'].apply(list).to_dict()
# 前 10
dbiansuxiang_top10 = dict(biansuxiang_dict1[:10])
print(dbiansuxiang_top10)
biansuxiang_top10 = list(dbiansuxiang_top10)
print(biansuxiang_top10)
biansuxiang_0 = biansuxiangdeal[biansuxiang_top10[0]]
biansuxiang_1 = biansuxiangdeal[biansuxiang_top10[1]]
biansuxiang_2 = biansuxiangdeal[biansuxiang_top10[2]]
biansuxiang_3 = biansuxiangdeal[biansuxiang_top10[3]]
biansuxiang_4 = biansuxiangdeal[biansuxiang_top10[4]]
biansuxiang_5 = biansuxiangdeal[biansuxiang_top10[5]]
biansuxiang_6 = biansuxiangdeal[biansuxiang_top10[6]]
biansuxiang_7 = biansuxiangdeal[biansuxiang_top10[7]]
biansuxiang_8 = biansuxiangdeal[biansuxiang_top10[8]]
biansuxiang_9 = biansuxiangdeal[biansuxiang_top10[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("top_1", biansuxiang_0)
    .add_yaxis("top_2", biansuxiang_1)
    .add_yaxis("top_3", biansuxiang_2)
    .add_yaxis("top_4", biansuxiang_3)
    .add_yaxis("top_5", biansuxiang_4)
    .add_yaxis("top_6", biansuxiang_5)
    .add_yaxis("top_7", biansuxiang_6)
    .add_yaxis("top_8", biansuxiang_7)
    .add_yaxis("top_9", biansuxiang_8)
    .add_yaxis("top_10", biansuxiang_9)
#     .set_global_opts(
#         title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#     )
    .render('./charts/变速箱销量前 10 销售周期散点图.html')
)
# 后 10
dbiansuxiang_dreciprocal10 = dict(biansuxiang_dict1[-10:])
print(dbiansuxiang_dreciprocal10)
# len(dbiansuxiang_reciprocal10)
biansuxiang_reciprocal10 = list(dbiansuxiang_dreciprocal10)
print(biansuxiang_reciprocal10)
biansuxiang_d0 = biansuxiangdeal[biansuxiang_reciprocal10[0]]
biansuxiang_d1 = biansuxiangdeal[biansuxiang_reciprocal10[1]]
biansuxiang_d2 = biansuxiangdeal[biansuxiang_reciprocal10[2]]
```

```
biansuxiang_d3 = biansuxiangdeal[biansuxiang_reciprocal10[3]]
biansuxiang_d4 = biansuxiangdeal[biansuxiang_reciprocal10[4]]
biansuxiang_d5 = biansuxiangdeal[biansuxiang_reciprocal10[5]]
biansuxiang_d6 = biansuxiangdeal[biansuxiang_reciprocal10[6]]
biansuxiang_d7 = biansuxiangdeal[biansuxiang_reciprocal10[7]]
biansuxiang_d8 = biansuxiangdeal[biansuxiang_reciprocal10[8]]
# biansuxiang_d9 = biansuxiangdeal[biansuxiang_reciprocal10[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("reci_1", biansuxiang_d0)
    .add_yaxis("reci_2", biansuxiang_d1)
    .add_yaxis("reci_3", biansuxiang_d2)
    .add_yaxis("reci_4", biansuxiang_d3)
    .add_yaxis("reci_5", biansuxiang_d4)
    .add_yaxis("reci_6", biansuxiang_d5)
    .add_yaxis("reci_7", biansuxiang_d6)
    .add_yaxis("reci_8", biansuxiang_d7)
    .add_yaxis("reci_9", biansuxiang_d8)
#       .add_yaxis("reci_10", biansuxiang_d9)
#       .set_global_opts(
#           title_opts = opts.TitleOpts(title = "后 10 的销售周期")
#       )
    .render('./charts/变速箱销量后 10 销售周期散点图.html')
)

# 燃油类型
# 排序，按燃油类型
ranyouleixing_dict = counter(list(df['燃油类型']))
# 绘制折线图显示
c = (
    Line()
    .add_xaxis(xaxis_data = list(ranyouleixing_dict))
    .add_yaxis("数量", y_axis = list(ranyouleixing_dict.values()))
    .render('./charts/燃油类型对销售周期的影响折线图.html')
)
# 按销量排序
ranyouleixing_dict1 = sorted(ranyouleixing_dict.items(), key = lambda x:x[1],
reverse = True)
# 整理燃油类型与成交周期
ranyouleixingdeal = df.groupby('燃油类型')['成交周期(天)'].apply(list).to_dict()
# 前 10
dranyouleixing_top10 = dict(ranyouleixing_dict1)
print(dranyouleixing_top10)
```

```
ranyouleixing_top10 = list(dranyouleixing_top10)
print(ranyouleixing_top10)
ranyouleixing_0 = ranyouleixingdeal[ranyouleixing_top10[0]]
ranyouleixing_1 = ranyouleixingdeal[ranyouleixing_top10[1]]
ranyouleixing_2 = ranyouleixingdeal[ranyouleixing_top10[2]]
ranyouleixing_3 = ranyouleixingdeal[ranyouleixing_top10[3]]
ranyouleixing_4 = ranyouleixingdeal[ranyouleixing_top10[4]]

c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", ranyouleixing_0)
        .add_yaxis("top_2", ranyouleixing_1)
        .add_yaxis("top_3", ranyouleixing_2)
        .add_yaxis("top_4", ranyouleixing_3)

#         .set_global_opts(
#                 title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#         )
        .render('./charts/燃油类型对销售周期影响散点图.html')
)

# 新车价—— 变化较大
# df['价格折损'] = df['新车价'] - df['上架价格']
# df_time = pd.concat([df['展销时间'], df['注册日期'], df['上牌日期'], df['上架时间'],
df['成交时间']], axis = 1)

# 基于皮尔逊相关系数
pearson1 = df.corr()
print(pearson1)

# 相关系数计算
df['价格差'] = df['新车价'] - df['上架价格']
df['上架时间'] = pd.to_datetime(df['上架时间'])
df['注册日期'] = pd.to_datetime(df['注册日期'])
df['日期差'] = df['上架时间'] - df['注册日期']
df['日期差'] = df['日期差'].apply(lambda x : x.days)
# 基于皮尔逊相关系数
pearson2 = df.corr()
print(pearson2)
xishu = dict(zip(list(pearson2.index), list(np.abs(np.array(list(pearson2['成交周期
(天)']))))))
xishu = sorted(xishu.items(), key = lambda x:x[1], reverse = True)
xishu = dict(xishu)
```

```
print(xishu)
```

2.2.3 销售周期区间分析

```
import pandas as pd
import numpy as np
from collections import Counter

from pyecharts import options as opts
from pyecharts.charts import Scatter
from pyecharts.charts import Line
from pyecharts.charts import Pie

def counter(list):
    from collections import Counter
    # 对列表内的元素进行排序，计数，以列表的形式返回
    list_counter = Counter(list)
    list_sorted = sorted(list_counter.items(), key = lambda x:x[0], reverse = False)
    # lambda x:x[1]按值排序，reverse = True 倒序 即大值在前。
    list_dict = dict(list_sorted)
    return list_dict

df = pd.read_csv('./data/附件 4-4（含周期区间）.csv')

# 成交周期
transaction = list(df['成交周期(天)'])
# transaction

# 成交周期区间
cycle_interval = list(df['周期标签'])
# print(type(cycle_interval[0]))

cycle_interval_dict0 = counter(cycle_interval)
cycle_interval_list = sorted(cycle_interval_dict0.items(), key = lambda x : x[1],
reverse = False)

cycleon = list(cycle_interval_dict0)

# 绘制销售周期分布图
c = (
    Pie()
    .add("", cycle_interval_list)
    .set_global_opts(title_opts = opts.TitleOpts(title = "销售周期区间分布饼图"))
    .set_series_opts(label_opts = opts.LabelOpts(formatter = "{b}:{c}"))
    .render('./charts/销售周期区间分布饼图.html')
)

# 分析品牌对销售周期区间的影响

brandid_dict = counter(list(df['品牌  id ']))
brandid_dict1 = sorted(brandid_dict.items(), key = lambda x:x[1], reverse = True)

brandid_top10 = dict(brandid_dict1[:10])

brand_top10 = list(brandid_top10)

brandcycle = df.groupby('品牌  id ')['周期标签'].apply(list).to_dict()

brand_0 = brandcycle[brand_top10[0]]
brand_1 = brandcycle[brand_top10[1]]
brand_2 = brandcycle[brand_top10[2]]
brand_3 = brandcycle[brand_top10[3]]
brand_4 = brandcycle[brand_top10[4]]
brand_5 = brandcycle[brand_top10[5]]
brand_6 = brandcycle[brand_top10[6]]
brand_7 = brandcycle[brand_top10[7]]
brand_8 = brandcycle[brand_top10[8]]
brand_9 = brandcycle[brand_top10[9]]

brand0 = list(counter(brand_0).values())
brand1 = list(counter(brand_1).values())
brand2 = list(counter(brand_2).values())
brand3 = list(counter(brand_3).values())
brand4 = list(counter(brand_4).values())
brand5 = list(counter(brand_5).values())
brand6 = list(counter(brand_6).values())
brand7 = list(counter(brand_7).values())
brand8 = list(counter(brand_8).values())
brand9 = list(counter(brand_9).values())

c = (
    Scatter()
    .add_xaxis(cycleon)
    .add_yaxis("top_1", brand0)
    .add_yaxis("top_2", brand1)
    .add_yaxis("top_3", brand2)
    .add_yaxis("top_4", brand3)
    .add_yaxis("top_5", brand4)
    .add_yaxis("top_6", brand5)
    .add_yaxis("top_7", brand6)
    .add_yaxis("top_8", brand7)
    .add_yaxis("top_9", brand8)
    .add_yaxis("top_10", brand9)
    #       .set_global_opts(
#           title_opts = opts.TitleOpts(title = "品牌前 10 的销售周期区间")
#       )
    .render('./charts/品牌前 10 的销售周期区间散点图.html')
)


# 车辆颜色
color_dict = counter(list(df['车辆颜色']))

colorlist = list(color_dict)

# 整理车辆颜色与成交周期区间
colorcycle = df.groupby('车辆颜色')['周期标签'].apply(list).to_dict()

color_0 = colorcycle[colorlist[0]]
color_1 = colorcycle[colorlist[1]]
color_2 = colorcycle[colorlist[2]]
color_3 = colorcycle[colorlist[3]]
color_4 = colorcycle[colorlist[4]]
color_5 = colorcycle[colorlist[5]]
color_6 = colorcycle[colorlist[6]]
color_7 = colorcycle[colorlist[7]]
color_8 = colorcycle[colorlist[8]]
color_9 = colorcycle[colorlist[9]]

color0 = list(counter(color_0).values())
color1 = list(counter(color_1).values())
color2 = list(counter(color_2).values())
color3 = list(counter(color_3).values())
color4 = list(counter(color_4).values())
color5 = list(counter(color_5).values())
color6 = list(counter(color_6).values())
color7 = list(counter(color_7).values())
color8 = list(counter(color_8).values())
color9 = list(counter(color_9).values())
```

```
c = (
        Scatter()
        .add_xaxis(cycleon)
        .add_yaxis("top_1", color0)
        .add_yaxis("top_2", color1)
        .add_yaxis("top_3", color2)
        .add_yaxis("top_4", color3)
        .add_yaxis("top_5", color4)
        .add_yaxis("top_6", color5)
        .add_yaxis("top_7", color6)
        .add_yaxis("top_8", color7)
        .add_yaxis("top_9", color8)
        .add_yaxis("top_10", color9)
#           .set_global_opts(
#                   title_opts = opts.TitleOpts(title = "前 10 的销售周期区间")
#           )
        .render('./charts/车辆颜色销售区间分布情况散点图.html')
)


# 过户次数

# 排序，按过户次数
guohucount_dict = counter(list(df['过户次数']))

guohucountlist = list(guohucount_dict)

# 整理过户次数与成交周期区间
guohucountcycle = df.groupby('过户次数')['周期标签'].apply(list).to_dict()

guohucount_0 = guohucountcycle[guohucountlist[0]]
guohucount_1 = guohucountcycle[guohucountlist[1]]
guohucount_2 = guohucountcycle[guohucountlist[2]]
guohucount_3 = guohucountcycle[guohucountlist[3]]
guohucount_4 = guohucountcycle[guohucountlist[4]]
guohucount_5 = guohucountcycle[guohucountlist[5]]
guohucount_6 = guohucountcycle[guohucountlist[6]]
guohucount_7 = guohucountcycle[guohucountlist[7]]
guohucount_8 = guohucountcycle[guohucountlist[8]]
guohucount_9 = guohucountcycle[guohucountlist[9]]

guohucount0 = list(counter(guohucount_0).values())
guohucount1 = list(counter(guohucount_1).values())
```

```
guohucount2 = list(counter(guohucount_2).values())
guohucount3 = list(counter(guohucount_3).values())
guohucount4 = list(counter(guohucount_4).values())
guohucount5 = list(counter(guohucount_5).values())
guohucount6 = list(counter(guohucount_6).values())
guohucount7 = list(counter(guohucount_7).values())
guohucount8 = list(counter(guohucount_8).values())
guohucount9 = list(counter(guohucount_9).values())


c = (
        Scatter()
        .add_xaxis(cycleon)
        .add_yaxis("top_1", guohucount0)
        .add_yaxis("top_2", guohucount1)
        .add_yaxis("top_3", guohucount2)
        .add_yaxis("top_4", guohucount3)
        .add_yaxis("top_5", guohucount4)
        .add_yaxis("top_6", guohucount5)
        .add_yaxis("top_7", guohucount6)
        .add_yaxis("top_8", guohucount7)
        .add_yaxis("top_9", guohucount8)
        .add_yaxis("top_10", guohucount9)
#          .set_global_opts(
#                  title_opts = opts.TitleOpts(title = "前 10 的销售周期区间")
#          )
        .render('./charts/过户次数销售区间分布情况散点图.html')
)



# 载客人数
# 排序，按载客人数
zaikerenshu_dict = counter(list(df['载客人数']))

zaikerenshulist = list(zaikerenshu_dict)

# 整理载客人数与成交周期区间
zaikerenshucycle = df.groupby('载客人数')['周期标签'].apply(list).to_dict()

zaikerenshu_0 = zaikerenshucycle[zaikerenshulist[0]]
zaikerenshu_1 = zaikerenshucycle[zaikerenshulist[1]]
zaikerenshu_2 = zaikerenshucycle[zaikerenshulist[2]]
zaikerenshu_3 = zaikerenshucycle[zaikerenshulist[3]]
```

```
zaikerenshu_4 = zaikerenshucycle[zaikerenshulist[4]]
zaikerenshu_5 = zaikerenshucycle[zaikerenshulist[5]]
zaikerenshu_6 = zaikerenshucycle[zaikerenshulist[6]]

zaikerenshu0 = list(counter(zaikerenshu_0).values())
zaikerenshu1 = list(counter(zaikerenshu_1).values())
zaikerenshu2 = list(counter(zaikerenshu_2).values())
zaikerenshu3 = list(counter(zaikerenshu_3).values())
zaikerenshu4 = list(counter(zaikerenshu_4).values())
zaikerenshu5 = list(counter(zaikerenshu_5).values())
zaikerenshu6 = list(counter(zaikerenshu_6).values())

c = (
    Scatter()
    .add_xaxis(cycleon)
    .add_yaxis("top_1", zaikerenshu0)
    .add_yaxis("top_2", zaikerenshu1)
    .add_yaxis("top_3", zaikerenshu2)
    .add_yaxis("top_4", zaikerenshu3)
    .add_yaxis("top_5", zaikerenshu4)
    .add_yaxis("top_6", zaikerenshu5)
    .add_yaxis("top_7", zaikerenshu6)
#     .set_global_opts(
#         title_opts = opts.TitleOpts(title = "前 10 的销售周期区间")
#     )
    .render('./charts/载客人数销售区间分布情况散点图.html')
)

# 燃油类型
# 排序，按燃油类型
ranyouleixing_dict = counter(list(df['燃油类型']))

ranyouleixinglist = list(ranyouleixing_dict)

# 整理燃油类型与成交周期区间
ranyouleixingcycle = df.groupby('燃油类型')['周期标签'].apply(list).to_dict()

ranyouleixing_0 = ranyouleixingcycle[ranyouleixinglist[0]]
ranyouleixing_1 = ranyouleixingcycle[ranyouleixinglist[1]]
ranyouleixing_2 = ranyouleixingcycle[ranyouleixinglist[2]]
ranyouleixing_3 = ranyouleixingcycle[ranyouleixinglist[3]]
ranyouleixing_4 = ranyouleixingcycle[ranyouleixinglist[4]]

ranyouleixing0 = list(counter(ranyouleixing_0).values())
```

```
ranyouleixing1 = list(counter(ranyouleixing_1).values())
ranyouleixing2 = list(counter(ranyouleixing_2).values())
ranyouleixing3 = list(counter(ranyouleixing_3).values())
ranyouleixing4 = list(counter(ranyouleixing_4).values())


c = (
    Scatter()
    .add_xaxis(cycleon)
    .add_yaxis("top_1", ranyouleixing0)
    .add_yaxis("top_2", ranyouleixing1)
    .add_yaxis("top_3", ranyouleixing2)
    .add_yaxis("top_4", ranyouleixing3)
#       .set_global_opts(
#           title_opts = opts.TitleOpts(title = "前 10 的销售周期区间")
#       )
    .render('./charts/燃油类型销售区间分布情况散点图.html')
)
```

2.2.4 销售周期区间分析 2

```
import pandas as pd
import numpy as np
from collections import Counter

from pyecharts import options as opts
from pyecharts.charts import Scatter
from pyecharts.charts import Line
from pyecharts.charts import Pie

def counter(list):
    from collections import Counter
    # 对列表内的元素进行排序，计数，以列表的形式返回
    list_counter = Counter(list)
    list_sorted = sorted(list_counter.items(), key = lambda x:x[0], reverse = False)
    # lambda x:x[1]按值排序，reverse = True 倒序 即大值在前。
    list_dict = dict(list_sorted)
    return list_dict

df = pd.read_csv('./data/附件 4-4（含周期区间）.csv')

# 成交周期
```

```
transaction = list(df['成交周期(天)'])

cycle = list(range(0, 278))

# 成交周期区间
cycle_interval = list(df['周期标签'])

cycle_interval_dict0 = counter(cycle_interval)
cycle_interval_list = sorted(cycle_interval_dict0.items(), key = lambda x : x[1],
reverse = False)
print(cycle_interval_list)
cycle_interval_dict1 = dict(cycle_interval_list)

cycleon = list(cycle_interval_dict0)

# 绘制销售周期分布图

# 绘制销售周期的散点图
c = (
    Scatter()
    .add_xaxis(list(cycle_interval_dict1))
    .add_yaxis(
        "次",
        list(cycle_interval_dict1.values()),
    )
    .set_global_opts(
        title_opts = opts.TitleOpts(title = '销售周期散点图')
    )
    .render('./charts/销售周期区间分布散点图 1.html')
)
c.render_notebook()

# 绘制销售周期的散点图
c = (
    Scatter()
    .add_xaxis(list(cycle))
    .add_yaxis(
        " ",
        list(cycle_interval),
    )
    .set_global_opts(
        title_opts = opts.TitleOpts(title = '销售周期散点图')
    )
)

# 分析品牌对销售周期区间的影响

brandid_dict = counter(list(df['品牌  id ']))

brandid_dict1 = sorted(brandid_dict.items(), key = lambda x:x[1], reverse = True)

brandid_top10 = dict(brandid_dict1[:10])

brand_top10 = list(brandid_top10)

brandcycle = df.groupby('品牌  id ')['周期标签'].apply(list).to_dict()

brand_0 = brandcycle[brand_top10[0]]
brand_1 = brandcycle[brand_top10[1]]
brand_2 = brandcycle[brand_top10[2]]
brand_3 = brandcycle[brand_top10[3]]
brand_4 = brandcycle[brand_top10[4]]
brand_5 = brandcycle[brand_top10[5]]
brand_6 = brandcycle[brand_top10[6]]
brand_7 = brandcycle[brand_top10[7]]
brand_8 = brandcycle[brand_top10[8]]
brand_9 = brandcycle[brand_top10[9]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", brand_0)
        .add_yaxis("top_2", brand_1)
        .add_yaxis("top_3", brand_2)
        .add_yaxis("top_4", brand_3)
        .add_yaxis("top_5", brand_4)
        .add_yaxis("top_6", brand_5)
        .add_yaxis("top_7", brand_6)
        .add_yaxis("top_8", brand_7)
        .add_yaxis("top_9", brand_8)
        .add_yaxis("top_10", brand_9)
#          .set_global_opts(
#                  title_opts = opts.TitleOpts(title = "品牌前 10 的销售周期区间")
#          )
        .render('./charts/品牌前 10 的销售周期区间散点图 1.html')
)
```

```
# 车辆颜色

color_dict = counter(list(df['车辆颜色']))

colorlist = list(color_dict)

# 整理车辆颜色与成交周期区间
colorcycle = df.groupby('车辆颜色')['周期标签'].apply(list).to_dict()

color_0 = colorcycle[colorlist[0]]
color_1 = colorcycle[colorlist[1]]
color_2 = colorcycle[colorlist[2]]
color_3 = colorcycle[colorlist[3]]
color_4 = colorcycle[colorlist[4]]
color_5 = colorcycle[colorlist[5]]
color_6 = colorcycle[colorlist[6]]
color_7 = colorcycle[colorlist[7]]
color_8 = colorcycle[colorlist[8]]
color_9 = colorcycle[colorlist[9]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", color_0)
        .add_yaxis("top_2", color_1)
        .add_yaxis("top_3", color_2)
        .add_yaxis("top_4", color_3)
        .add_yaxis("top_5", color_4)
        .add_yaxis("top_6", color_5)
        .add_yaxis("top_7", color_6)
        .add_yaxis("top_8", color_7)
        .add_yaxis("top_9", color_8)
        .add_yaxis("top_10", color_9)
#         .set_global_opts(
#                 title_opts = opts.TitleOpts(title = "前 10 的销售周期区间")
#         )
        .render('./charts/车辆颜色销售区间分布情况散点图 1.html')
)


# 过户次数

# 排序，按过户次数
guohucount_dict = counter(list(df['过户次数']))
```

```
guohucountlist = list(guohucount_dict)

# 整理过户次数与成交周期区间
guohucountcycle = df.groupby('过户次数')['周期标签'].apply(list).to_dict()


guohucount_0 = guohucountcycle[guohucountlist[0]]
guohucount_1 = guohucountcycle[guohucountlist[1]]
guohucount_2 = guohucountcycle[guohucountlist[2]]
guohucount_3 = guohucountcycle[guohucountlist[3]]
guohucount_4 = guohucountcycle[guohucountlist[4]]
guohucount_5 = guohucountcycle[guohucountlist[5]]
guohucount_6 = guohucountcycle[guohucountlist[6]]
guohucount_7 = guohucountcycle[guohucountlist[7]]
guohucount_8 = guohucountcycle[guohucountlist[8]]
guohucount_9 = guohucountcycle[guohucountlist[9]]
c = (
    Scatter()
    .add_xaxis(cycle)
    .add_yaxis("top_1", guohucount_0)
    .add_yaxis("top_2", guohucount_1)
    .add_yaxis("top_3", guohucount_2)
    .add_yaxis("top_4", guohucount_3)
    .add_yaxis("top_5", guohucount_4)
    .add_yaxis("top_6", guohucount_5)
    .add_yaxis("top_7", guohucount_6)
    .add_yaxis("top_8", guohucount_7)
    .add_yaxis("top_9", guohucount_8)
    .add_yaxis("top_10", guohucount_9)
#       .set_global_opts(
#           title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#       )
    .render('./charts/过户次数销售区间分布情况散点图 1.html')
)



# 载客人数
# 排序，按载客人数
zaikerenshu_dict = counter(list(df['载客人数']))
zaikerenshulist = list(zaikerenshu_dict)

# 整理载客人数与成交周期区间
zaikerenshucycle = df.groupby('载客人数')['周期标签'].apply(list).to_dict()
```

```
# zaikerenshucycle

zaikerenshu_0 = zaikerenshucycle[zaikerenshulist[0]]
zaikerenshu_1 = zaikerenshucycle[zaikerenshulist[1]]
zaikerenshu_2 = zaikerenshucycle[zaikerenshulist[2]]
zaikerenshu_3 = zaikerenshucycle[zaikerenshulist[3]]
zaikerenshu_4 = zaikerenshucycle[zaikerenshulist[4]]
zaikerenshu_5 = zaikerenshucycle[zaikerenshulist[5]]
zaikerenshu_6 = zaikerenshucycle[zaikerenshulist[6]]
c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", zaikerenshu_0)
        .add_yaxis("top_2", zaikerenshu_1)
        .add_yaxis("top_3", zaikerenshu_2)
        .add_yaxis("top_4", zaikerenshu_3)
        .add_yaxis("top_5", zaikerenshu_4)
        .add_yaxis("top_6", zaikerenshu_5)
        .add_yaxis("top_7", zaikerenshu_6)
#           .set_global_opts(
#                   title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#           )
        .render('./charts/载客人数销售区间分布情况散点图 1.html')
)



# 燃油类型
# 排序，按燃油类型
ranyouleixing_dict = counter(list(df['燃油类型']))

ranyouleixinglist = list(ranyouleixing_dict)

# 整理燃油类型与成交周期区间
ranyouleixingcycle = df.groupby('燃油类型')['周期标签'].apply(list).to_dict()

ranyouleixing_0 = ranyouleixingcycle[ranyouleixinglist[0]]
ranyouleixing_1 = ranyouleixingcycle[ranyouleixinglist[1]]
ranyouleixing_2 = ranyouleixingcycle[ranyouleixinglist[2]]
ranyouleixing_3 = ranyouleixingcycle[ranyouleixinglist[3]]
ranyouleixing_4 = ranyouleixingcycle[ranyouleixinglist[4]]

c = (
        Scatter()
        .add_xaxis(cycle)
        .add_yaxis("top_1", ranyouleixing_0)
        .add_yaxis("top_2", ranyouleixing_1)
        .add_yaxis("top_3", ranyouleixing_2)
        .add_yaxis("top_4", ranyouleixing_3)
        .add_yaxis("top_5", ranyouleixing_4)

#         .set_global_opts(
#                 title_opts = opts.TitleOpts(title = "前 10 的销售周期")
#         )
        .render('./charts/燃油类型销售区间分布情况散点图 1.html')
)
```

2.3 进一步挖掘

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
data = pd.read_csv(r"C:\Users\Lenovo\Desktop\总.csv")
df = pd.DataFrame(data)
print(df.corr())
print(df.corr('spearman'))
print(df.corr('kendall'))
hm = plt.subplots(figsize = (10,10))
hm = sns.heatmap(data.corr(), square=True, annot=True)
plt.show()


data = pd.read_csv("C:\\Users\Lenovo\Desktop\\4.csv", encoding='utf-8')
#print(data)
for i in data.updatePriceTimeJson:
        i = i.split("")
        if len(i) > 1:
          print(i)
          print(len(i))
          print(i[1])
          data.updateTime = data.updatePriceTimeJson.apply(lambda x:i[1])
          data.updatePrice = i[3]

print(data)


data = pd.read_csv(r"C:\Users\Lenovo\Desktop\4_all.csv")
data.withdrawDate = data.withdrawDate.astype(str)
data.withdraw_month = data.withdrawDate.apply(lambda x:x[5:6])
print(data)
data.to_csv("C:\\Users\Lenovo\Desktop\\4_all2.csv", encoding='utf-8')



data = pd.read_csv(r"C:\\Users\Lenovo\Desktop\\4_all2.csv", encoding='utf-8')
print(Counter(data.withdraw_month))
```

2.4 问题二模型

2.4.1 仅附件四模型

```
import pandas as pd
import numpy as np
import re
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor

df_1=pd.read_table(r'D:\比赛\2021 年 MathorCup 大数据竞赛赛道 A\附件\附件
4：门店交易训练数据.txt',sep='\t',

parse_dates=[1,4,5],header=None,names=['carid','pushDate','pushPrice','updateP
riceTimeJson'
                        ,'pullDate','withdrawDate'])
data=pd.DataFrame(data=df_1)
data['pushDate_year']=data['pushDate'].dt.year
data['pushDate_month']=data['pushDate'].dt.month
data['pushDate_day']=data['pushDate'].dt.day

data['withdrawDate_year']=data['withdrawDate'].dt.year
data['withdrawDate_month']=data['withdrawDate'].dt.month
data['withdrawDate_day']=data['withdrawDate'].dt.day


data_1=data
data_1['period']=data_1['withdrawDate']-data_1['pushDate']
# pd.to_datetime(data['period'])     dtype timedelta64[ns] cannot be converted to
datetime64[ns]
data_1['period']=data_1['period'].apply(lambda x: str(x).split(' ')[0])
data_1['period']=data_1['period'].apply(lambda x: int(x) if x!='NaT' else np.nan)

data_1['updatePriceTimeJson'].astype('string')
def deal(data=None):
    li=[]
    for i in data.index:
            count=len(re.findall('\d+-\d+-\d+',str(data.loc[i,'updatePriceTimeJson'])))
            li.append(count)

        return li

data_1['decreasing_count']=deal(data_1)

data_1.info()

data_2=data_1.drop(['pushDate','pullDate','withdrawDate','updatePriceTimeJson'],
axis=1)
data_2=data_2.dropna()

y_=data_2['period']
data_2=data_2.drop(['period'],axis=1)
data_2['period']=y_
data_2.info()


pearson=data_2.corr()
index = pearson['period'][:-1].abs() > 0.01
X=data_2.iloc[:,:-1]
X_submit=X.loc[:,index]
X_submit.columns
X_train,X_test,y_train,y_test=train_test_split(X_submit,data_2['period'],test_size=
0.2)

random_model=RandomForestRegressor(n_estimators=300,random_state=33,n
_jobs=-1)
random_model.fit(X_train,y_train)
y_pred=random_model.predict(X_test)
random_model.score(X_test,y_test)
```

2.4.2 结合附件四和附件一模型

```
import pandas as pd
import numpy as np
import re
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor


# df = pd.read_table('../data/附件 1：估价训练数据.txt',sep='\t',encoding='gbk',
#                               parse_dates=[1,11,12],header=None)
# df_1 = pd.read_table('../data/ 附 件 4 ： 门 店 交 易 训 练 数
据.txt',sep='\t',encoding='gbk',
#                                    parse_dates=[1,4,5],header=None)
df = pd.read_table(r'D:\比赛\2021 年 MathorCup 大数据竞赛赛道 A\附件\附件 1：
估价训练数据.txt',sep='\t',encoding='gbk',
                             parse_dates=[1,11,12],header=None)
df_1=pd.read_table(r'D:\比赛\2021 年 MathorCup 大数据竞赛赛道 A\附件\附件
4：门店交易训练数据.txt',sep='\t',

parse_dates=[1,4,5],header=None,names=['carid','pushDate','pushPrice','updateP
riceTimeJson'
                         ,'pullDate','withdrawDate'])


def add_columns(data=None):
      columns = ['carid', 'tradeTime', 'brand', 'serial', 'model', 'mileage', 'color',
'cityId', 'carCode',
                       'transferCount', 'seatings', 'registerDate', 'licenseDate', 'country',
'maketype', 'modelyear',
                       'displacement', 'gearbox', 'oiltype', 'newprice']
      for i in range(1, 16):
            str_ = 'anonymousFeature' + str(i)
            columns.append(str_)
      columns.append('price')
      data.columns = columns
      return data
df=add_columns(df)
#
df_1.rename(columns={0:'carid',1:'pushDate',2:'pushPrice',3:'updatePriceTimeJso
n',4:'pullDate',5:'withdrawDate'},inplace=True)
data = pd.merge(df,df_1,how='right')
data.info()
data=data.drop(['country','anonymousFeature4','anonymousFeature7','anonymous
Feature10','anonymousFeature15'
                         ,'maketype','anonymousFeature1','anonymousFeature8','anony
mousFeature9'],axis=1)
data.info()
nn = ['carCode', 'modelyear', 'gearbox']
for i in nn:
      x = int(data[i].mode())
      data[i].fillna(x, inplace=True)
      #data[i]=data[i].dropna()

"""
特征工程
```

```
"""
data['tradeTime_year']=data['tradeTime'].dt.year
data['tradeTime_month']=data['tradeTime'].dt.month
data['tradeTime_day']=data['tradeTime'].dt.day
data['registerDate_year']=data['registerDate'].dt.year
data['registerDate_month']=data['registerDate'].dt.month
data['registerDate_day']=data['registerDate'].dt.day


# 这个特征不适用与预测周期
data['old_year'] = data['tradeTime'] - data['registerDate']
data['old_year'] = data['old_year'].apply(lambda x: str(x).split(' ')[0])
data['old_year'] = data['old_year'].astype(int)

# 处理匿名特征 11
data_1=data.dropna()                          #
data_1.info()
len(list(data_1['anonymousFeature11']))
# data_1['anonymousFeature11'].fillna('0',inplace=True)
def deal_11(x):
     return sum([float(x) for x in re.findall("\d",x)])
data_1['anonymousFeature11']=data_1['anonymousFeature11'].map(deal_11)
data_1.info()
# 处理匿名特征 12
def deal_12(x):
     li=[float(x) for x in re.findall("\d+",x)]
     return li[0]*li[1]*li[2]
data_1['anonymousFeature12_length']=data_1['anonymousFeature12'].apply(lam
bda x:int(x.split('*')[0]))
data_1['anonymousFeature12_width']=data_1['anonymousFeature12'].apply(lam
bda x:int(x.split('*')[1]))
data_1['anonymousFeature12_height']=data_1['anonymousFeature12'].apply(lam
bda x:int(x.split('*')[2]))
data_1['anonymousFeature12']=data_1['anonymousFeature12'].map(deal_12)
# 处理匿名特征 13
def deal_13(x):
     return x[:4], x[4:6]
data_1['anonymousFeature13']=data_1['anonymousFeature13'].astype('string')
data_1['anonymousFeature13_year']=data_1['anonymousFeature13'].map(deal_1
3)      #   (2017, 09)
data_1['anonymousFeature13_month']=data_1['anonymousFeature13_year'].appl
y(lambda x: int(x[1]))
data_1['anonymousFeature13_year']=data_1['anonymousFeature13_year'].apply(l
ambda x: int(x[0]))
```

```
data_1['anonymousFeature13']=data_1['anonymousFeature13'].astype('float')


data_1['old_year_1']=data_1['tradeTime']-data_1['licenseDate']
data_1['old_year_1']=data_1['old_year_1'].apply(lambda x:str(x).split(' ')[0])
data_1['old_year_1']=data_1['old_year_1'].astype(int)


data_1['period']=data_1['withdrawDate']-data_1['pushDate']
# pd.to_datetime(data['period'])     dtype timedelta64[ns] cannot be converted to datetime64[ns]
data_1['period']=data_1['period'].apply(lambda x: str(x).split(' ')[0])
data_1['period']=data_1['period'].apply(lambda x: int(x) if x!='NaT' else np.nan)


"""
特征处理
"""
data_new = data_1.dropna(subset=['period'])
data_new.info()
data_new.isnull().sum()

data_new_2=data_new.dropna()
data_new_2.info()

#  处理 updatePriceTimeJson
data_new_2['updatePriceTimeJson'].astype('string')
def deal(data=None):
    li=[]
    for i in data.index:
        count=len(re.findall('\d+-\d+-\d+',str(data.loc[i,'updatePriceTimeJson'])))
        li.append(count)

    return li

data_new_2['decreasing_count']=deal(data_new_2)
data_new_2.info()
data_new_3=data_new_2.select_dtypes(include=['float64', 'int32','int64'])

y_=data_new_3['period']
data_new_3=data_new_3.drop(['period'],axis=1)
data_new_3['period']=y_
data_new_3.info()
```

```
# data_new_3.to_csv(r'D:\比赛\2021 年 MathorCup 大数据竞赛赛道 A\任务二
result.csv')

pearson=data_new_3.corr()
index=pearson['price'][:-1].abs() > 0.1
X=data_new_3.iloc[:,:-1]
X_subset = X.loc[:, index]





X_train,X_test,y_train,y_test=train_test_split(data_new_3.iloc[:,:-1],data_new_3['p
eriod'])


random_model=RandomForestRegressor(n_estimators=300,random_state=33,n
_jobs=-1)
random_model.fit(X_train,y_train)
y_pred=random_model.predict(X_test)

random_model.score(X_test,y_test)
random_model.score(X_train,y_train)


# float("inf") 无穷大
```