基于灰色评价模型和集成学习算法的电动汽车营销策略

摘要

本文研究电动汽车销售的影响因素和营销策略，通过建立灰色关联度评价模型以分析电动汽车综合实力，运用 Local Cascade Ensemble (LCE) 算法预测目标客户购买情况，建立收益-风险规划模型得到有针对性的营销策略。

针对问题一，进行异常值和缺失值的数据清洗，建立**灰色关联度评价模型**，对不同品牌电动汽车综合实力进行分析。利用**主成分分析**和 **K-means++** 对数据进行**降维聚类**，得到每个类型对应人群，再利用该人群指标数据的平均值来代替其中的**异常值**。通过**逻辑关系判断**对**缺失值**进行补全。提取**客户体验感**、**可靠性**、**偏向性**三个中层指标，及八种满意度打分对应的**平均值**、**标准差系数**、**偏度**共 24 个底层指标，建立评价不同品牌综合实力的**灰色关联度评价模型**，并通过**改进的熵权法**求得指标权重，求出**综合实力**得分。三种品牌综合实力得分分别为：**45.03**、**76.97**、**32.03**。

针对问题二，基于 4 种**集成学习方法**得出指标数据的**特征重要性**，进行**特征选择**后得出不同品牌电动汽车对应的重要指标。根据定类数据的逻辑性特点，运用**标签编码**和**哑变量编码**，对指标数据中的定类数据进行定量转化。利用 **GBDT**、**CatBoost**、**XGBoost** 和**随机森林** 4 种集成学习方法度量特征重要性，通过测试集的 **F1 分数**得出 4 种方法所占权重，按权重计算出各指标的综合特征重要性，选出针对**不同品牌**的重要指标。合资品牌重要指标为 **a1**、**a3**、**B11**、**B13**、**B14**、**B16**、**B17**；自主品牌重要指标为 **a1**、**a3**、**a4**、**a5**、**B15**、**B16**、**B17**；新势力品牌重要指标为 **a1**、**B1**、**B3**、**B6**、**B16**、**B17**。

针对问题三，利用 **Local Cascade Ensemble (LCE) 算法**，结合各品牌的重要指标，预测客户购买情况。本文建立基于 **LCE 算法**的预测模型，结合不同品牌对应的重要指标数据，对客户购买结果进行预测，并与 **XGBoost** 以及**随机森林**方法进行比较，利用**准确率**和 **AUC** 评价该模型的**优良性**。之后，再利用该模型对 15 名目标客户的购买情况进行判断，得出有购买行为的对象为**客户 1**、**客户 6 和客户 12**，它们的购买概率分别为 **0.9429**、**0.9927**、**0.6342**。

针对问题四，建立**收益-风险规划模型**，针对不同品牌挑选目标客户并实施营销策略。首先建立**客户筛选模型**对目标客户群体进行初步的筛选，再据此引入**风险度**，结合已有指标数据信息，建立**收益-风险规划模型**，针对不同品牌挑选 1 名没有购买电动汽车目标客户并实施营销策略。最终得到客户 5、客户 7、客户 12 的提升后购买概率为 **64.35%**、**59.36%**、**63.42%**。

针对问题五，本文根据前四问的结论，对该企业销售部门提出销售策略建议。

本文的优势在于： 1. 利用主成分分析和 K-means++ 对数据进行分组讨论处理异常值，体现了模型的准确性和逻辑严密性；2. 引入新型算法 LCE，提高求解精度。

关键字：**主成分分析　集成学习　LCE 算法　收益-风险规划模型**

目录

一、 问题重述

1.1 问题背景

汽车产业一直在国民经济中占据十分重要的地位，为我国经济发展提供积极动力。而新能源汽车产业作为新型战略产业，在汽车产业中的重要性不容忽视。我国已经把发展新能源汽车作为国家战略。新能源汽车不仅是全球汽车产业转型升级的重要偏向，也是应对能源危机、减缓情况污染的重要抓手 [6]。

汽车产业是国民经济的重要支柱产业，而新能源汽车产业是战略性新兴产业。无论是从解决能源环境问题的角度还是国家政策支持的角度来看，电动汽车产业都有着良好的发展前景。但是，电动汽车毕竟是一个新兴的事物，与传统汽车相比，消费者在某些方面还存在着一些疑虑，因此其市场销售需要科学决策。

图 1　电动汽车产业图

1.2 问题提出

近年来，电动汽车产业快速发展，有着广阔的市场前景。但电动汽车产业仍是新兴产业，客户在选择过程中仍存在顾虑，因此市场营销策略的科学性就显得尤为重要。某公司推出了三种品牌的电动汽车，并进行了客户购买意愿调查，从而制定相应的营销策略。针对这些客户对不同品牌满意度信息及个人特征，我们对以下几个问题进行研究：

问题一：对题目提供的客户对不同品牌满意度信息及个人特征数据进行数据清洗，然后对数据进行描述性统计分析。

问题二：结合客户对不同品牌满意度信息及个人特征，以及购买了体验电动汽车的客户信息，研究对不同品牌电动汽车的销售有影响的因素。

问题三：结合之前得出的结果，建立针对不同品牌电动汽车的客户挖掘模型，并对模型优良性进行评价。

问题四：销售部门认为，只要加大营销力度，客户满意度就能在短期内得到提高。基于这种思路和之前的研究结果，针对每个品牌挑选出 1 名没有购买电动汽车的目标客户，实施销售策略。

问题五：根据之前的研究结果，给销售部门提出销售策略建议。

二、 模型假设

（1）附件已知数据真实可靠，不存在统计误差等情况；

（2）时间为同一维度，客户的满意度和个人特征不会被时间的波动所影响；

（3）各项数据指标间相关性较弱。

三、 符号说明

符号	含义	说明
$X_{1,i}$	满意度打分平均值	$i=1,2,\cdots,8$
$X_{2,i}$	标准差系数	$i=1,2,\cdots,8$
$X_{2,i}$	偏度	$i=1,2,\cdots,8$
$m_{i,j}$	第 i 个方案中的第 j 个评价指标	$i=1,2,3;\ j=1,2,\cdots,n$
p_{ij}	第 j 项评价指标下第 i 个方案占该指标比重	$i=1,2,3;\ j=1,2,\cdots,n$

四、 问题一的模型建立与求解

4.1 问题一的描述与分析

问题一要求对题目提供的客户对不同品牌满意度信息及个人特征数据进行数据清洗，然后对数据进行描述性统计分析。首先，我们对**异常值**和**缺失数据**进行处理。本文对指标进行**降维聚类**得到每个类型对应人群，再利用该人群指标数据的平均值来代替其中的异常值。对于缺失数据，本文通过**逻辑关系判断**对其进行补全。

在对数据进行描述性统计分析时，为了对目标客户对于不同品牌汽车满意度的比较分析，本文建立**灰色关联评价模型**对不同品牌汽车的综合实力进行打分。本文选择**客户体验感**、**可靠性**、**偏向性**作为一级指标，八组满意度得分平均值、标准差系数和偏度分

别作为客户体验感、可靠性、偏向性对映的二级指标，各层指标分别使用**改进的熵权法**确定权重。

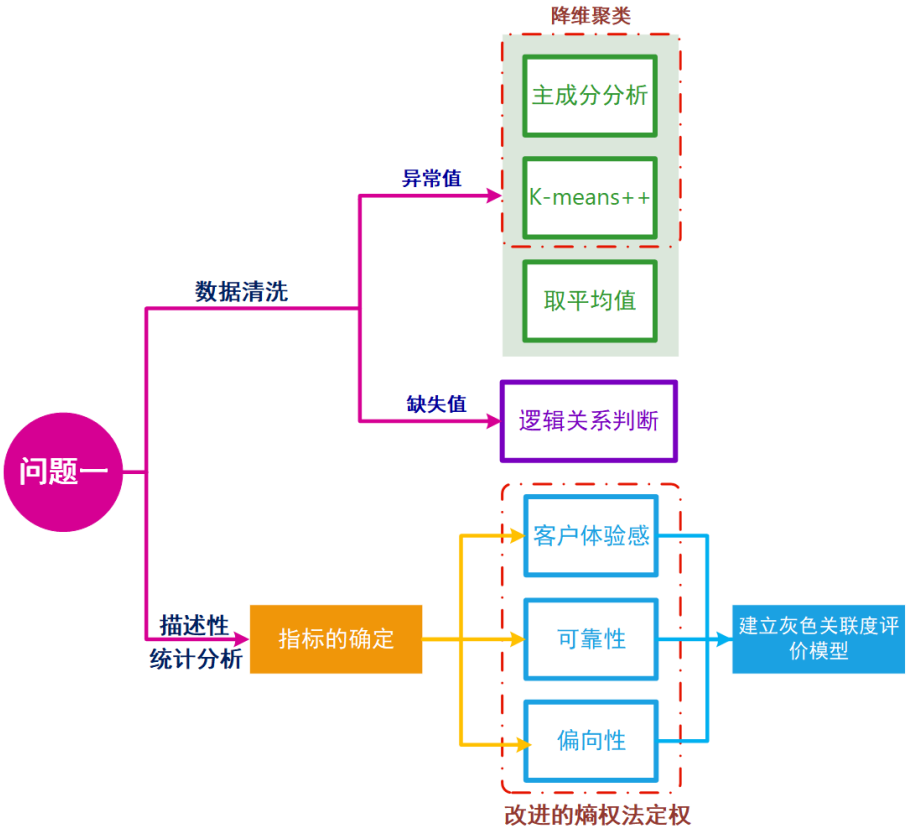

图 2　问题一思路图

4.2 数据清洗工作

在对各指标数据进行描述性统计分析之前，本文首先对数据进行清洗工作，即对异常值和缺失数据进行处理。对于异常值，本文利用主成分分析法对数据进行降维后，再用 K-means++ 对客户数据进行聚类处理，并用该异常值对应客户类别的指标平均值代替该异常值。对于缺失值，本文通过对客户个人特征的逻辑关系判断对其进行补全。

4.2.1 异常值处理

从逻辑性的角度出发分析各指标数据，我们发现客户的满意度打分中存在超过 100 分的异常值情况。从数据可靠性的角度出发，我们发现客户个人特征中的占比数据存在超过 100% 的情况，与数据特质不符。此外，对各指标数据进行 3σ 检验发现，其中还存在 64 个离群值，也会对分析结果造成影响。

由于客户的满意度打分和客户个人特征都具有较强的独特性，若通过直接对数据求平均的方法对异常值进行处理，可能会造成分析结果的不准确。因此，本文首先通过聚类的方法将客户人群进行分类，确定该异常值所属客户的对应人群，然后对该人群对应

指标进行求平均值来代替异常值。然而，由于满意度打分以及客户个人特征均有多个数据指标，而在高维数据的情况下数据点的距离倾向于彼此接近，数据点之间的距离关系会弱化。因此本文选择首先利用主成分分析法对数据进行降维，再利用 K-means++ 进行聚类分析 [4]。

- **主成分分析**

主成分分析是把原来多个变量划为少数几个综合指标的一种统计分析方法。由于本问题中指标数量较多，对于有异常值数据的指标，我们对其余七项指标进行主成分分析，将数据维度降至 2 维。若异常值出现在 $a1$ 指标的数据中，则对 $a2, a3, \cdots, a8$ 这七个指标进行主成分分析，分析结果如下表1:

表 1　主成分分析结果表

特征向量	a2	a3	a4	a5	a6	a7	a8	特征值	累计贡献率
r1	-0.37	-0.393	-0.369	-0.384	-0.379	-0.366	-0.384	437.35	0.7895
r2	0.057	-0.891	0.112	0.007	0.233	0.241	0.28	37.35	0.8569
r3	...	...	...	...	...	...	...	...	...

从上表可以看出，前一个和前两个主成分的累计贡献率分别达到 78.95% 和 85.69%，第一主成分 r1 在所有变量上都有近似相等的负载荷，反映了客户对该品牌汽车总体体验情况，因此第一主成分可称为综合客户体验成分。第二主成分 r2 在变量 a2 上有很高的正载荷，而在其余变量上有很小的正载荷。可以认为这个主成分度量了受汽车品牌经济性影响的消客户体验感，第二主成分可称为经济性倾向成分。而第三主成分很难给出明显的解释，因此我们只取前面两个主成分。

此外，对于异常值出现在其他指标中的情况，与此种情况使用同样方法，最终均确定将数据指标降至二维。

- **K-Means++ 聚类**

K-means++ 算法在选择聚类中心时采用原则是保证初始聚类中心之间距离尽可能远，从而可以对 K-means 算法进行改进。本文在利用主成分分析法对数据指标进行降维后，选择 K-means++ 算法对数据进行聚类，从而确定异常值所属类别。

进行 K-means++ 算法进行聚类时，首先输入聚类个数 k 和最大迭代次数 n，随机抽取样本为聚类中心。计算样本与已知中心的距离，距离越大越有可能是聚类中心。对 k 个聚类中心进行初始化，分配各列数据对象到距离最近的类中。若收敛或达到最大迭代次数，则可认为聚类以达到期望的结果，算法终止，否则继续重复之前步骤。

针对本文 K-means++ 聚类的 k 值确定，本文采用肘部法进行确定。肘部函数基于成本函数即各类畸变程度之和进行计算。对 a2-a8 指标的主成分 $r1$、$r2$，本文绘制聚类的肘部图如图3所示，其横坐标为聚类类别数 k，纵坐标为畸变程度。由图可见，当类数从 1 增加到 4 时，总畸变程度下降较快；但当类别数超过 4 时，总畸变程度变化趋势变缓。对此，本文确定 k=4 为最佳聚类数。主成分 $r1$、$r2$ 按照 k=4 进行聚类后，聚类效果散点图展示如图4。

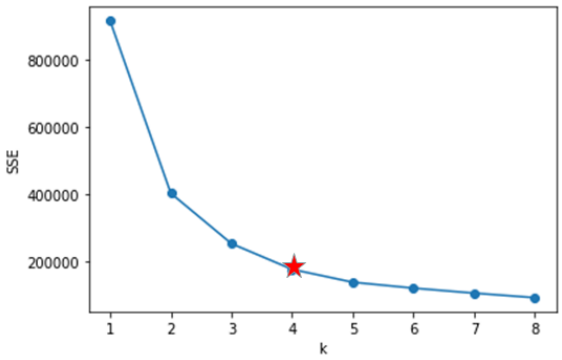

图 3　聚类肘部图

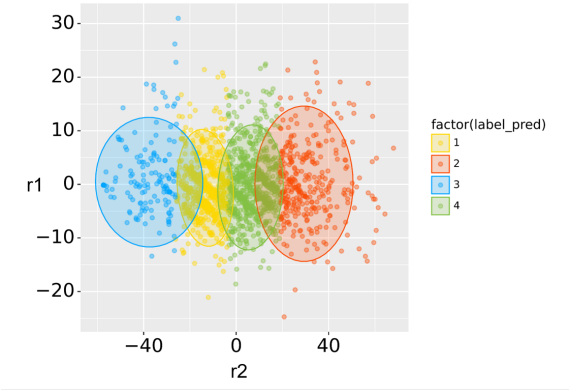

图 4　聚类效果散点图

对于异常值出现在其他指标中的情况，与上文使用同样聚类方法。

- **取平均值**

在利用主成分分析法对数据降维以及 K-means++ 聚类后，对异常数据所属客户类别的数据进行求平均值处理，则利用该平均值对指标异常值进行替代。

4.2.2 缺失数据处理

本文通过对客户个人特征的逻辑关系判断对缺失值进行补全。分析客户满意度打分以及客户个人特征相关数据发现，客户个人特征相关数据中的 B7 这一指标数据存在缺失的情况。因此，我们对该指标进行逻辑关系判断补全。由于 B7 为该客户子女数量，基于我国社会现状以及常见家庭结构，我们可以通过对该客户其他个人特征指标家庭成员人数 B5、婚姻家庭情况 B6 等指标对该客户子女数量进行逻辑判断。缺失数据的 3 种类型及处理方式如下：

1. 该缺失数据对应客户的婚姻家庭情况为“未婚”或“已婚无子女”，即 B6 对应数据为 1、2、3、4 时，本文将该客户子女数量认定为 0。

2. 该缺失数据对应客户的婚姻家庭情况为“已婚，有小孩，不与父母同住”，即 B6 对应数据为 5 时，结合该客户的家庭成员人数又有以下 2 种情况：

（1）家中有两口人：配偶在外务工，夫妻其中一方带子女独居的情况为我国一些地区的普遍社会现象。因此，这种情况下该客户的子女数量为 1 人。

（2）家中有四口人：若不与父母同住且家中有四口人，本文默认该家庭结构完整，即夫妻双方均在家居住，则该客户子女数量为 2 人。

3. 该缺失数据对应客户的婚姻家庭情况为“离异/丧偶”，即 B6 对应数据为 7 时，结合缺失数据对应客户的出生年份可知，若该客户有孩子且与孩子居住，其孩子未达到法定工作年龄，即不具备经济能力。因此，我们可以结合该客户家庭收入与个人收入对子女数量进行判断：

（1）若家庭收入大于个人收入，则说明该客户与父母同住，即可根据家庭成员人数得出该客户子女数量。

（2）若家庭收入等于个人收入，则说明该客户未与父母同住，同样可根据家庭成员人数得出该客户子女数量。

4.3 预备工作

4.3.1 不同类型品牌综合实力评价指标选取

为了更好地对不同类型品牌的电动汽车进行评估，本文选择不同品牌电动汽车的综合实力进行分析和比较，即建立多层评价模型对综合实力进行打分。结合八项满意度打分指标的数据特征，本文针对综合实力选择了客户体验感、可靠性、偏向性作为一级指标进行评价打分。

本文选择客户数据中的八项满意度得分的平均值作为客户体验感的二级指标，选择标准差系数作为二级指标对可靠性进行衡量，选择偏度作为偏向性的二级指标。各指标的意义及计算方法如下：

- **平均值 $X_{1,i}$ ($i = 1, 2, \cdots, 8$)**

本文选择客户数据中的八项满意度得分的平均值作为客户体验感的二级指标。满意度得分平均值满足下式：

$$X_{1,i} = \frac{\sum_{j=1}^{n} x_{ij}}{n} \quad (i = 1, 2, \cdots, 8). \tag{1}$$

其中，x_{ij} 表示第 i 个满意度得分指标的第 j 个数据结果，n 表示该满意度得分指标对应的数据数量。

- **标准差系数 $X_{2,i}$ ($i = 1, 2, \cdots, 8$)**

标准差系数是从相对角度观察的差异和离散程度，在比较相关事物的差异程度时较之直接比较标准差要好些。标准差系数计算公式如下：

$$X_{2,i} = \frac{\sigma_i}{X_{1,i}} \quad (i = 1, 2, \cdots, 8). \tag{2}$$

其中，σ_i 表示第 i 个满意度得分指标数据的标准差。

- **偏度 $X_{3,i}$ $(i = 1, 2, \cdots, 8)$**

偏度是统计数据分布偏斜方向程度的度量，其计算公式如下：

$$X_{3,i} = \frac{\mu_3}{\sigma_i^3}. \tag{3}$$

其中，μ_3 表示该数据的三阶中心距，σ_i 表示第 i 个满意度得分指标数据的标准差。

4.3.2 数据标准化处理

为方便使用灰色关联度评价模型对不同品牌电动汽车综合实力进行评价打分，本文对上述八项满意度指标对应的满意度打分平均值、标准差系数、偏度这 24 个评价指标进行标准化处理，从而避免量纲差异对评价结果的影响。标准化公式如下式4：

$$x^* = \frac{x_{ij} - \min x_{ij}}{\max x_{ij} - \min x_{ij}}. \tag{4}$$

4.4 模型建立

4.4.1 多层评价目标树建立

为对不同品牌电动汽车的综合实力进行评价打分，本文建立多层评价目标树从多个方面对不同品牌电动车的综合实力进行全面评价 [11]。其中，总目标为综合实力得分，中间目标为客户体验感、可靠性和偏向性。其中，客户体验感对应的二级评价指标是八项客户满意度打分平均值的数据指标，可靠性对应的二级评价指标是八项客户满意度打分标准差系数的数据指标，偏向性对应的二级评价指标是八项客户满意度打分偏度的数据指标。具体多层评价目标树如图5：

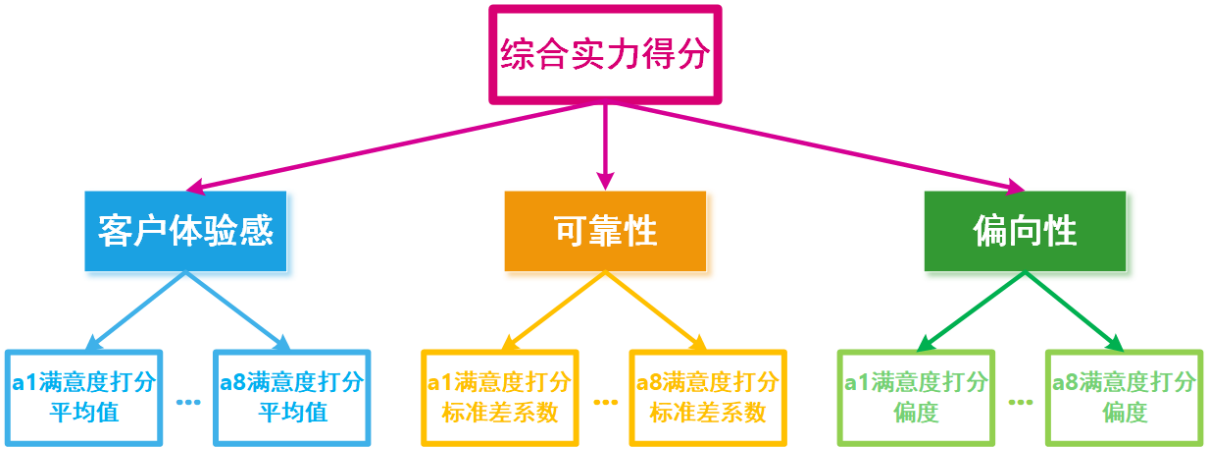

图 5　多层目标评价树

4.4.2 改进的熵权法为评价指标赋权值

熵权法的基本思路是根据指标变异性的大小确定客观权重，某个指标的信息熵越小，则说明该指标的变异程度越大，即它提供的信息量也越多，权重也就越大。而传统熵权法在所有熵值趋近于 1 时，会过度放大权重差距，赋权不合理。由于指标差距越小权重越大，本文对熵权法中的权重计算进行改进，从而克服传统赋权确定的不合理性。

Step1：建立初始评价矩阵和指标数据标准化。针对本题评价客户体验感，我们有 3 个评价方案，即 3 种不同品牌的电动汽车，8 个评价指标，即 8 项客户满意度打分平均值的数据指标。$m_{i,j}(i=1,2,3;j=1,2,\cdots,8)$ 是第 i 个方案中第 j 个评价指标，则初始评价矩阵为 $M=(m_{i,j})_{3\times 8}$。由于各项指标量纲和单位不统一，因此对矩阵 M 进行标准化处理，得到标准化矩阵 $T=(t_{i,j})_{3\times 8}$。

由于评价人口属性时所用指标均为正向指标，即选用指标的值越大越好，因此按照式 (4) 进行标准化：

$$t_{i,j}=\frac{m_{i,j}-m_{\min}}{m_{\max}-m_{\min}}. \tag{5}$$

Step2：计算各项指标的信息熵。第 j 项指标的信息熵为：

$$p_{i,j}=\frac{t_{i,j}}{\sum_{i=1}^{3}t_{i,j}}. \tag{6}$$

$$e_j=-\frac{1}{\ln 3}\sum_{i=1}^{n}p_{i,j}\ln p_{i,j}. \tag{7}$$

其中，$p_{i,j}$ 为第 j 项指标下第 i 个方案占该指标的比重。

Step3：确定各指标权重。第 j 项指标的权重为：

$$v_j=\begin{cases}(1-\overline{e}^{\alpha})v_j+\overline{e}^{\alpha}\overline{v_j}, & e_j<1\\ 0, & e_j=1\end{cases}. \tag{8}$$

其中，$\overline{v_j}$ 为指标 m_j 的平均权重，e_j 为对应的熵值，$\overline{e}$ 为全部不为 1 的熵权的平均值，α 为熵值指数，一般取评价方案的个数。

针对可靠性和偏向性，我们均有 3 个评价方案，即 3 个不同品牌电动汽车，8 个评价指标，即八项满意度打分对应的标准差系数或偏度的数据指标。对于这两个中间目标，本文同样使用上述改进的熵权法计算过程对这些指标赋权值。

通过改进的熵权法计算得到不同品牌汽车客户体验感、可靠性、偏向性的二级评价指标和中间目标权重如图6和图7所示：

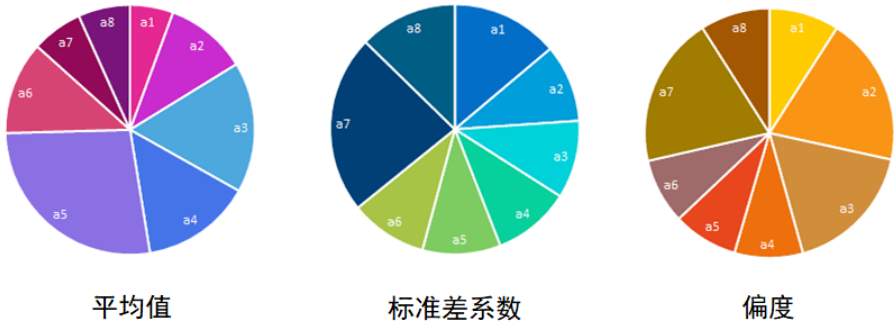

图 6　底层指标权重图

图 7　中间目标权重

4.4.3 灰色关联度模型为中层指标评价打分

灰色关联分析通常用于分析各个因素对结果的影响程度，以及解决综合评价类问题。本文利用灰色关联度模型联系各项指标数据对综合实力进行评价打分。根据灰色关联决策的理论，以评价方案指标向量与相对最优方案指标向量的关联度作为评价方案优劣的准则。

针对本题评价客户体验感、可靠性、偏向性的各个评价指标，其相对最优方案均为每个指标的最小值 $u_0 = (f_{01}, f_{02}, f_{03})$，规范化处理后为 $u_0 = (1, 1, 1)$，设计方案 u_i 的评价指标 v_j 与相对最优方案 u_0 的评价指标 v_j 之间的灰色关联度为：

$$\gamma_i = \frac{\xi \max |f_{ij} - 1|}{|f_{ij} - 1| + \xi \max |f_{ij} - 1|}. \tag{9}$$

其中，$\xi \in (0.1)$ 为分辨系数，一般取作 0.5。

设得到的灰色关联度矩阵分别为 γ_1、γ_2 和 γ_3，其客户体验感的评价指标权重向量为 W_1，可靠性评价指标权重向量为 W_2，偏向性评价指标权重向量为 W_3，则不同品牌电动汽车客户体验感打分为 $\gamma_1 W_1^T$，可靠性打分为 $\gamma_2 W_2^T$，偏向性打分为 $\gamma_3 W_3^T$.

4.4.4 不同品牌电动汽车综合实力量化分析

与对中间目标的评价打分类似，本文利用灰色关联度模型和改进的熵权法，结合客户体验感、可靠性、偏向性对不同品牌电动汽车的综合实力进行打分，设打分结果为

$Q = q_1, q_2, q_3$。

4.5 模型求解

结合模型建立，将模型建立的各个步骤进行求解计算，其整体的求解步骤如下：

Step1：利用改进的熵权法客户体验感为下的八个评价指标——八项客户满意度打分平均值的数据指标，可靠性下的八个评价指标——八项客户满意度打分标准差系数的数据指标，偏向性下的八个评价指标——八项客户满意度打分偏度的数据指标，权重向量分别为 W_1、W_2、W_3。

Step2：利用灰色关联度模型分别求出客户体验感下的八个评价指标、可靠性下的八个评价指标、偏向性下的八个评价指标的灰色关联度矩阵 γ_1、γ_2 和 γ_3，不同品牌电动汽车的客户体验感、可靠性、偏向性可计算为 $\gamma_1 W_1^T$、$\gamma_2 W_2^T$ 和 $\gamma_3 W_3^T$。

Step3：利用灰色关联度模型和熵权法，结合客户体验感、可靠性和偏向性对综合实力进行打分，之后对数据进行等级量化处理使其具有现实意义，据此得到不同品牌电动汽车的综合实力。

4.6 求解结果与分析

在进行了异常值以及缺失值的替换和补全后，我们各项指标数据进行了描述性统计量的分析，部分指标分析结果如下表2。由满意度打分指标的各个统计量可知，客户对不同满意度指标的打分情况基本类似，同一统计量之间差别较小。而对于客户个人特质指标数据所得的统计量，我们发现不同指标具有较大差异性。其他满意度打分以及客户个人特征指标的描述性统计量见附录。

表 2　部分指标描述性统计量展示表

描述统计量	a1	a2	a3	B2	B4	B5
标准误差	0.200706	0.195371	0.2359	0.258239	0.093147	0.024388
中位数	77.80869	77.80347	77.80532	20	7	3
峰度	1.366412	0.302209	0.951054	-1.17569	1.395267	-0.04528
最小值	33.15991	51.04517	29.588	1	1	1
最大值	99.03689	99.03017	99.03255	60	30	6
求和	153049.5	153754.5	149067.2	41997	14998	6787

此外，利用各指标数据绘制 a1-a8 指标如图8。由该箱线图可以看出，经过数据清洗后的各指标数据离群值不再出现较大的偏移，数据分布较为均匀且波动性较小。

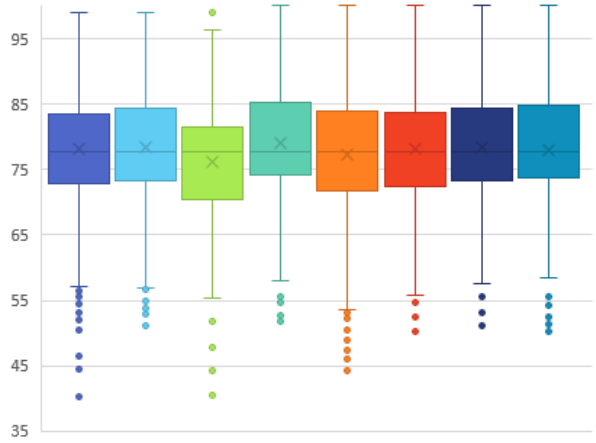

图 8　a1-a8 指标数据箱线图

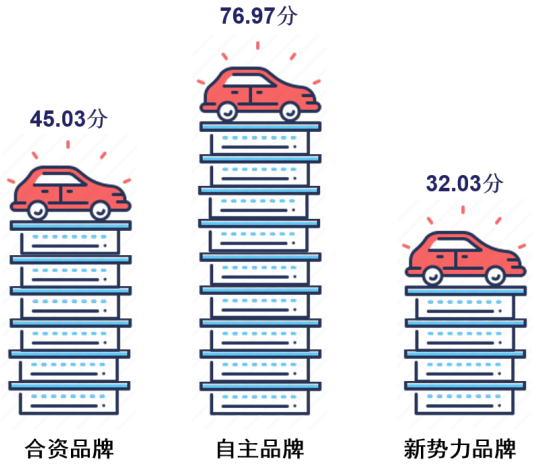

图 9　三种品牌电动汽车综合实力得分

本文利用灰色关联度评价模型和改进的熵权法，求解得到不同类型电动汽车品牌的综合实力得分展示如图9。在品牌类型方面，合资品牌和新势力品牌正在加速进入新能源汽车市场，与自主品牌形成鼎足之势。观察三种品牌电动汽车的综合实力得分可以发现，自主品牌电动汽车的综合实力得分高于合资品牌和新势力品牌。因此，合资品牌与新势力品牌若想在市场中形成竞争优势，还需要更完善的市场营销策略与企业发展规划。

结合世界电动汽车行业的发展现状对自主品牌的领先优势进行分析，我们发现全球电动汽车销量第一的企业比亚迪自主品牌占比最大，该事实也印证了自主品牌的压倒性优势。综合来看，随着新能源汽车市场的不断完善与发展，越来越多的企业明确了市场定位，推出了高质量的产品，市场整体的用户满意度也在稳步上升，并且电动汽车市场不同品牌间呈现良性竞争的市场状态。

五、 问题二的模型建立与求解

5.1 问题二的描述与分析

问题二要求结合客户对不同品牌满意度的信息及个人特征，以及体验并购买了电动汽车的客户信息，研究对不同品牌电动汽车的销售有影响的因素。本文针对不同品牌分别进行基于特征重要性的特征选择，选出影响不同品牌电动汽车销售的指标。其中，本文结合梯度提升决策树 **GBDT**、**CatBoost**、**XGBoost** 和**随机森林** 4 种集成学习的方法，进行特征的重要性度量。针对不同电动汽车品牌，根据不同方法的 **F1 分数**得出 4 种方法所占**权重**，综合得出特征重要性，选出针对不同品牌的重要指标。

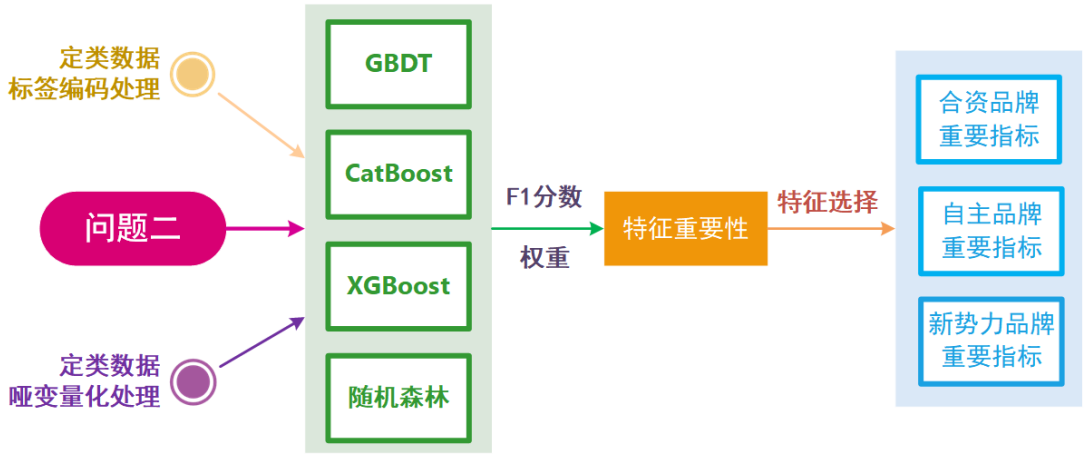

图 10　问题二思路图

5.2 处理定类数据

对于客户个人特征指标数据中的定类数据，我们需要将其转化为定量数据进行后续分析。对于定类数据中数据关联性大或存在逻辑关系的数据，我们采用标签编码的方式对其进行数值赋值转化为定量数据。对于其他定类数据，我们采用哑变量编码的方法对其进行转化。

- **哑变量编码**

哑变量化指的是对离散的特征进行编码，把某一个分类型变量各个值对应的权重都增加某一数值，同时把另一个分类型变量各个值对应的权重都减小某一数值。本文对指标数据中没有逻辑关系的定类数据进行哑变量化处理，将其转化为定量数据。与热编码相比，哑变量编码降低了数据的多重共线性，更适用于本文。对于 B1、B6、B11 的指标数据，本文选用哑变量编码方式进行转化。

- **标签编码**

对于定类数据中定序类型的数据，即内部存在关联性与逻辑关系的数据，本文通过标签编码自定义数字顺序，不破坏数据原有逻辑。对于其余客户个人特征指标数据，本文选用标签编码的方式对其进行转化。

5.3 模型建立

为了得出影响不同品牌电动汽车销售的，本文结合梯度提升决策树 GBDT、CatBoost、XGBoost 和随机森林 4 种集成学习的方法，进行特征的重要性度量。针对不同电动汽车品牌，根据不同方法的 F1 分数得出 4 种方法所占权重，综合得出特征重要性，选出针对不同品牌的重要指标 [9]。

5.3.1 特征重要性计算

首先，利用以下四种集成学习方法针对不同品牌特征重要性进行计算。

- **梯度提升决策树 GBDT**

梯度提升决策树 GBDT(Gradient Boosting Decision Tree) 是一种迭代的决策树算法，由多棵决策树组成 [5]。在使用 GBDT 进行特征重要性度量时，特征 j 的全局重要度通过特征 j 在单颗树中的重要度的平均值来衡量：

$$\hat{J}_j^2 = \frac{1}{N}\sum_{m=1}^{N}\hat{J}_j^2(T_m). \tag{10}$$

其中，N 是树的数量。

特征 j 在单颗树中的重要度的如下：

$$\hat{J}_j^2(T) = \sum_{t=1}^{L-1}\hat{i}_t^2 1(v_t = j). \tag{11}$$

其中，L 为树的叶子节点数量，$L-1$ 即为树的非叶子节点数量（构建的树都是具有左右孩子的二叉树），v_t 是和节点 t 相关联的特征，$\hat{i_t}$ 是节点分裂之后平方损失的减少值。

- **CatBoost**

CatBoost 与其它梯度提升算法类似，它每构建一颗新树，都会近似当前模型的梯度。但所有经典的 boosting 算法都存在有偏的逐点梯度估计带来的过拟合问题，因此 CatBoost 试图修正梯度偏差以缓解过拟合。解决方法为对数据集先随机排序，对于每个样本的该类别特征中的某个取值，转换为数值型时都是基于该样本之前的类别 label value 取均值，同时加入了优先级（先验值）和优先级（先验值）的权重系数。

- **XGBoost**

XGBoost 与 GBDT 方法相同，都是基于提升树的思想建立的。XGBoost 在 GBDT 的基础上引入了二阶导数和正则化。XGBoost 在进行特种重要性度量时，根据结构分数的增益情况计算出来选择哪个特征作为分割点，而某个特征的重要性就是它在所有树中出现的次数之和。也就是说一个属性越多的被用来在模型中构建决策树，它的重要性就相对越高。

- **随机森林**

利用随机森林进行特征的重要性度量，选择重要性较高的特征，计算某个特征 R 的重要性时，具体步骤如下 [3]：

（1）对每一颗决策树，选择相应的袋外数据（out of bag，OOB）计算袋外数据误差，记为 errOOB1.

（2）随机对袋外数据 OOB 所有样本的特征 R 加入噪声干扰（可以随机改变样本在特征 R 处的值），再次计算袋外数据误差，记为 errOOB2.

（3）假设森林中有 N 棵树，则特征 R 的重要性为 $\sigma(errOOB2 - errOOB1)/N$。

5.3.2 特征选择

特征选择的步骤如下：

（1）计算每个特征的重要性，并按降序排序；

（2）确定要剔除的比例，依据特征重要性剔除相应比例的特征，得到一个新的特征集；

（3）用新的特征集重复上述过程，直到剩下 m 个特征（m 为针对不同品牌数据情况设定的值）；

（4）根据上述过程中得到的各个特征集和特征集对应的袋外误差率，选择袋外误差率较低的特征集。

5.3.3 特征重要性度量

根据四种集成学习方法测试集的 F1 分数计算得出四种方法所占权重从而得到综合的特征重要性。F1 分数是统计学中用来衡量二分类模型精确度的一种指标，它可以看作是模型精确率和召回率的一种调和平均。通过下式来计算四种方法对应的权重 L_i：

$$L_i = \frac{F1_i}{\sum_{i=1}^{4} F1_i}. \tag{12}$$

其中 $F1_i$ 为四种集成学习方法的 F1 分数。

5.4 模型求解

结合模型建立，将模型建立的各个步骤进行求解计算，其整体的求解步骤如下：

Step1：通过 GBDT、CatBoost、XGBoost、随机森林 4 种集成学习方法计算各指标特征重要性。

Step2：通过 4 种方法的 F1 分数得出各个方法所占权重，再根据该权重对 Step1 中得到的特征重要性进行综合计算，得到各指标的最终特征重要性。

Step3：利用各指标特征重要性度量结果进行特征选择，得到对不同品牌电动汽车销售有影响的特征指标。

5.5 求解结果与分析

首先，根据 GBDT、CatBoost、XGBoost 和随机森林得出的测试集 F1 分数计算针对三个品牌的权重分布如下表3。由表中数据可知，不同品牌在进行特征选取时，GBDT、CatBoost、XGBoost 和随机森林这四种方法所占权重均匀，体现了综合算出的特征重要性结果的可靠性。

表 3　不同品牌 4 种方法权值表

	GBDT	CatBoost	XGBoost	随机森林
合资品牌	0.247559	0.249934	0.25099	0.251518
自主品牌	0.248378	0.25253	0.251752	0.24734
新势力品牌	0.253555	0.246149	0.253555	0.246742

利用个指标特征重要性度量结果进行特征选择，得到不同品牌电动汽车对应的特征指标体现如下图11。其中，合资品牌重要指标为 a1、a3、B11、B13、B14、B16、B17；自主品牌重要指标为 a1、a3、a4、a5、B15、B16、B17；新势力品牌重要指标为 a1、B1、B3、B6、B16、B17。

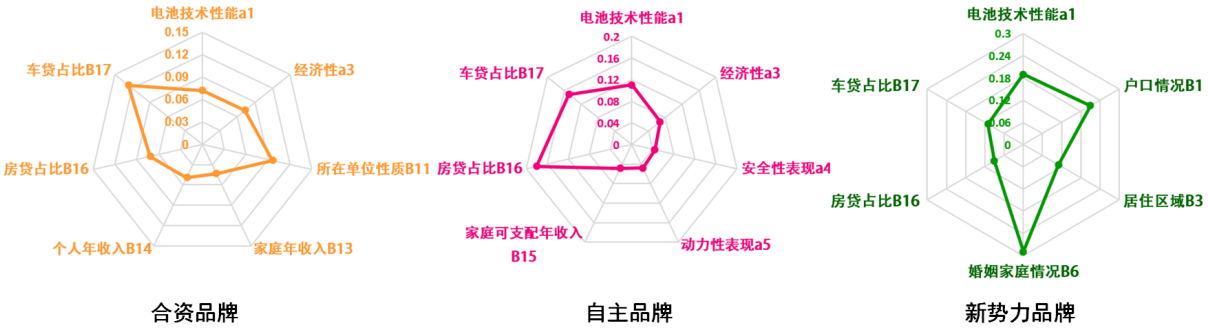

图 11　不同品牌重要指标

分析合资品牌结果，发现对该品牌销售情况具有较大影响的指标为车贷占比 B17 和客户所在单位性质 B11。对于自主品牌结果，其重要指标为房贷占比 B16 和车贷占比 B17。而对于新势力品牌，则是婚姻家庭情况 B6 与户口情况 B1 对其影响最大。综合分析发现，电池技术性能 a1、房贷占比 B16、车贷占比 B17 这三个重要指标为这三种品牌所共有的。根据以上结论，我们发现电动汽车本身的质量以及客户的经济状况往往会决定该客户的购买行为。

六、 问题三的模型建立与求解

6.1 问题三的描述与分析

问题三要求结合之前得出的结果，建立针对不同品牌电动汽车的客户挖掘模型，并对模型优良性进行评价。本文使用 **Local Cascade Ensemble (LCE) 算法**建立预测模型，结合问题二中得到的各品牌对应的重要指标，对该客户是否购买该品牌汽车的结果进行预测，对比其他方法的预测效果，对模型优良性进行评价 [8]。

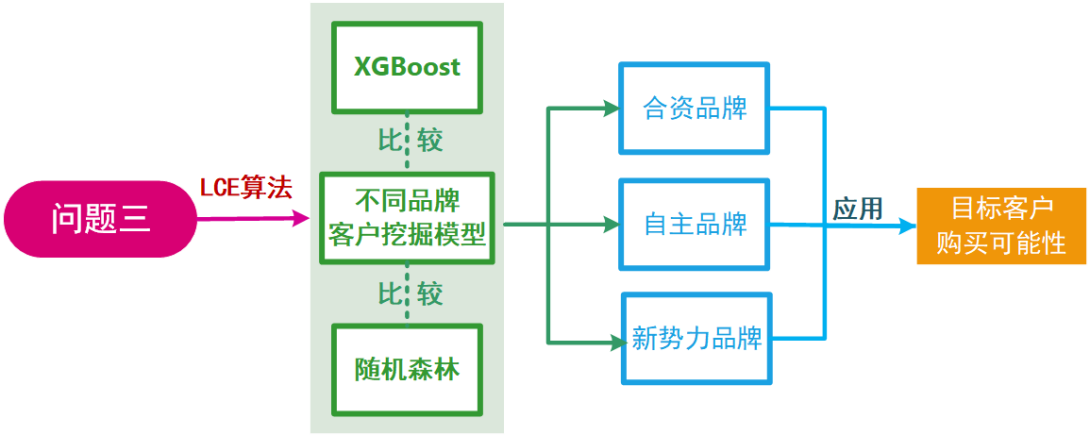

图 12　问题三思路图

6.2 缺失数据处理

观察附件 3 中的目标客户信息数据发现，客户子女数量 B7 的数据存在缺失，此处本文通过问题一中建立的逻辑关系判断对缺失值进行补全。

6.3 LCE 算法预测模型建立

Local Cascade Ensemble (LCE) 算法是一种新的集成学习方法，它结合了随机森林和 XGBoost 算法的优势，并采用互补的多样化方法来获得更好的预测性能。

我们通常通过对偏差与方差的评估来对该算法进行评价。Bagging 方法通过生成多个版本的预测器并聚合预测器，从而减少算法的方差。Boosting 方法通过迭代学习弱预测器并相加以创建最终强预测器，从而减少算法的偏差。随机森林和 XGBoost 分别是 Bagging 和 Boosting 方法目前最先进的算法。而 LCE 算法将这两种算法进行结合，以达到同时最小化偏差和方差的效果 [2]。

结合问题二中得到的重要性指标数据，本文选择 LCE 算法对客户购买不同品牌电动汽车情况进行预测，将预测结果与实际购买情况进行对比得出预测效果，并继续利用该算法对 15 名目标客户的购买对应品牌电动汽车的可能性进行判断。

6.4 模型求解

结合问题二中得到的重要性指标数据，利用 LCE 算法对客户购买不同品牌电动汽车情况进行预测求解。

LCE 采用分而治之的方法来个性化训练数据不同部分的预测误差。针对本题对 LCE 算法的应用，本文通过 LCE 算法预测购买情况的求解过程如图13。

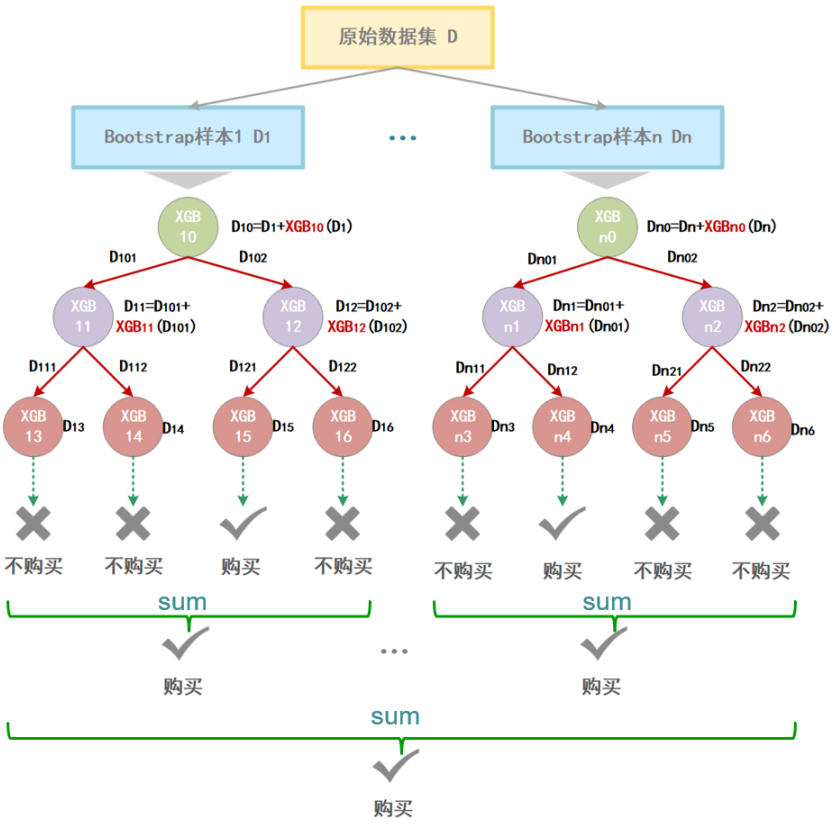

图 13　LCE 算法示意图

在生成树的过程中，将每个决策节点处的基学习器的输出作为新属性添加到数据集来沿树向下传播提升，如图中的 $XGB_{10}(D_1)$。在下一个树级别，添加到数据集的输出被基础学习器用作加权方案，这样可以更多的关注先前错误预测的样本。最后，Bagging 通过从随机抽样中创建多个预测变量并替换原始数据集以简单多数票聚合树来降低方差，如图中的 D_1、D_2.

此外，在利用 LCE 算法对购买行为进行判断的过程中，当购买概率大于 50% 即判断该客户有购买行为时，预测结果最接近真实情况，因此确定阈值为 0.5.

6.5 求解结果与分析

首先，本文使用 Local Cascade Ensemble (LCE) 算法建立预测模型，结合问题二中得到的各品牌对应的重要指标，对该客户是否购买该品牌汽车的结果进行预测，对比其他方法的预测效果，对模型优良性进行评价。然后，再利用 LCE 算法预测模型结合品牌种类对 15 位目标客户的购买情况进行判断。

6.5.1 模型优良性评价

将 LCE 算法与 XGBoost 以及随机森林的预测效果进行比较，比较结果如图14。由图可以看出 LCE 算法对应 ROC 曲线下方 AUC 面积最接近 1，说明基于 LCE 算法的预

测模型性能良好，且优于 XGBoost 与随机森林的方法。

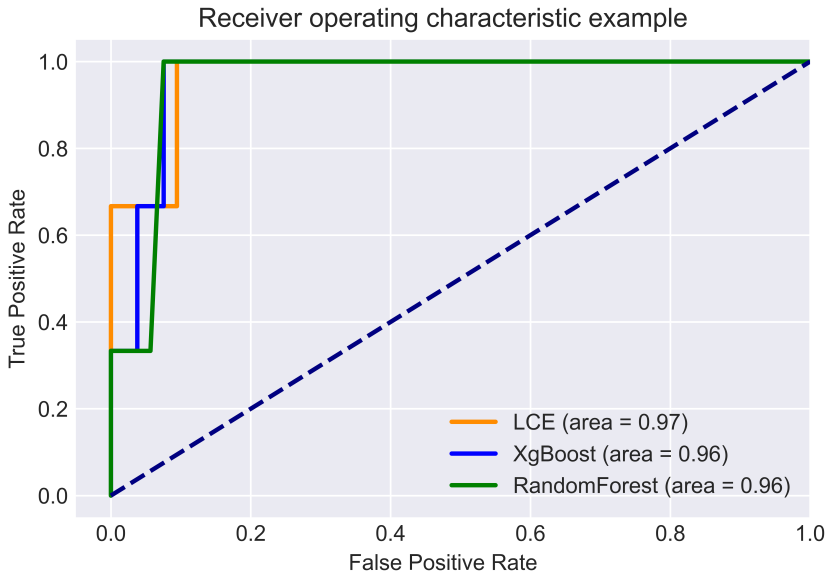

图 14　三种方法对应 ROC 曲线

通过 LCE 算法、XGBoost 和随机森林算法的预测得到三种方法对应的准确度、敏感性、精确率以及 F1 值如下表4。由表可知，LCE 算法的准确度、敏感性、精确率以及 F1 值均大于 XGBoost 和随机森林方法，由此可以证明本文所用的 LCE 算法预测模型的优良性，使文章的求解结果更可靠。

表 4　三种方法效果比较

	准确度	敏感性	精确率	F1 值
LCE	0.98	0.982321	0.981071	0.979821
XGBoost	0.95	0.944464	0.954286	0.948571
随机森林	0.95	0.945179	0.935357	0.939464

6.5.2 目标客户购买情况判断

通过已知信息训练得到针对三种品牌的 LCE 算法后，利用 LCE 算法预测模型结合品牌种类对 15 位目标客户的购买情况进行判断。预测结果如下表5。

表 5　目标客户购买概率表

客户编号	品牌编号	购买概率	是否购买	客户编号	品牌编号	购买概率	是否购买
1	1	0.9429	1	9	2	0.0287	0
2	1	0.0011	0	10	2	0.0288	0
3	1	0.0005	0	11	3	0.6342	1
4	1	0.0006	0	12	3	0.3873	0
5	1	0.4425	0	13	3	0.0729	0
6	2	0.9927	1	14	3	0.0062	0
7	2	0.3504	0	15	3	0.1167	0
8	2	0.0287	0				

购买概率大于 0.5 时，本文认定该目标客户有购买行为。

分析表中数据可知，客户 1、客户 6、客户 11 有极大可能购买电动汽车，而客户 3、客户 4、客户 12、客户 13、客户 14 几乎没购买的可能。总体来看，大部分客户的购买意愿并不大，因此企业还需要加大宣传以及服务力度，改善电动汽车销售现状。

七、 问题四的模型建立与求解

7.1 问题四的描述与分析

问题四要求基于短期内加大服务力度能提高客户满意度的思路和之前的研究结果，针对每个品牌挑选出 1 名没有购买电动汽车的目标客户，实施销售策略。对此本文首先建立**客户筛选模型**对目标客户群体进行初步的筛选，再据此引入**风险度**，结合已有指标数据信息，建立**收益-风险规划模型**，针对不同品牌挑选目标客户并实施营销策略 [12]。

7.2 客户筛选模型建立

为了针对不同的品牌选出最合适的目标客户，本文首先对目标客户群体进行初步筛选。若目标客户 a1-a8 的满意度指标均提高五个百分点后，利用 LCE 算法对该客户的购买行为仍然判断为否，则说明短期内对该客户的服务力度投入并不会得到回报。因此，本文建立客户筛选模型，初步筛选得到满意度指标均提高五个百分点后会购买电动汽车的目标客户。基于问题三中确定的阈值 0.5，得到客户筛选条件如下：

$$P_0 + P > 50\%. \tag{13}$$

其中，P_0 为目标客户的初始购买可能性，P 为投入服务后目标客户购买可能性的变化量。

7.3 收益-风险规划模型建立

本问题的目的是对客户实施相应的销售策略，本模型针对不同品牌，筛选得到相应的潜在客户并对客户进行最优的服务策略。

目标函数

首先，本文基于客户筛选模型引入风险度 E，即客户满意度提高后该客户仍不购买的行为对企业投资造成了一定的风险。因此，本文建立目标函数使得企业服务投入的风险最小：

$$\min E = [1 - (P_0 + P)] \times \lambda. \tag{14}$$

其中，E 为企业服务投入风险，P_0 为目标客户的初始购买可能性，P 为投入服务后目标客户购买可能性的变化量，λ 为体现客户不购买行为与企业投入关系的风险系数。

由题知，营销者服务难度与提高的满意度百分点是成正比的，设服务难度与提高满意度百分比的比例为难度系数 k。基于对客户购买行为的分析以及相关文献 [10]，得到难度系数 k 与购买概率 P_0 满足如下关系：

$$k = ln(\theta P_0 + e). \tag{15}$$

其中，θ 为边际成本系数，e 自然常数。

除企业投入风险最小的目标外，企业通过该投入行为获得最大收益，结合服务难度与客户增加的各项满意度百分比，本文建立目标函数时企业收益 M 最大：

$$\max M = \xi P - \zeta k \sum_{i=1}^{8} x_i^2. \tag{16}$$

其中，M 为企业收益，ξ 为售价参考系数，ζ 为成本参考系数，x_i 为 LCE 中所需的重要指标的增加百分点。

综上所述，结合风险 E 和收益 M，得到收益-风险规划模型的总目标函数为：

$$\max Z = P_0(M, -E). \tag{17}$$

约束条件

由题可知八项客户满意度指标提升百分点不得大于 5%，则有：

$$x_i \leq 5\% \quad . \tag{18}$$

结合相关参考文献 [1] 对风险度的描述，我们对于风险 E 确定以下约束：

$$E \leq 36\% \tag{19}$$

综上所述，该收益-风险规划模型建立如下：

$$\max Z = P_0(M, -E).$$

$$\begin{cases} x_i \leq 5\%, \\ E \leq 36\%, \\ k = ln(\theta P_0 + e), \\ M = \xi P - \zeta k \sum_{i=1}^{8} x_i^2, \\ \min E = [1 - (P_0 + P)] \times \lambda. \end{cases} \tag{20}$$

7.4 模型求解

通过客户筛选模型，可以得到三种品牌中值得实施销售策略的目标客户为客户 5、客户 7、客户 12。以客户 5 为例，进行收益-风险模型规划求解。若品牌追求效益最大化，则最大风险为 54.98%，若品牌追求风险最小化，则其获得最大效益为 64.32%。通过对风险值的约束，可以得到收益与风险的关系如下表6。

表 6　收益与风险的关系表

风险	收益	a1 增加百分点	a3 增加百分点
53%	48.90%	0.0099	0.015
51%	52.10%	0.0215	0.0232
49%	63.90%	0.02989	0.02822
47%	50.60%	0.02532	0.2939

分析上表，可以发现因为收益会受到成本的影响，随风险的减少呈现出先上升后下降的趋势。综合考虑以上因素和结果，本文以 30% 为权重构建出以下目标模型：

$$\max Z = M - \omega E. \tag{21}$$

7.5 求解结果与分析

客户 5 所属的品牌类型为合资品牌，对于 a1 和 a3 的提升比例为 35.27% 和 31.81%。客户 7 所属的品牌类型为自主品牌，指标为 a1、a3、a4、a5，提升的比例分别为 19.01%、23.77%、20.92%、19.97%。客户 12 所属的品牌类型为新势力品牌，指标 a1 的提升比例

为 18.85%。下图15为客户 5、客户 7 和客户 12 的原购买电动汽车概率与规划求解后的提升概率。

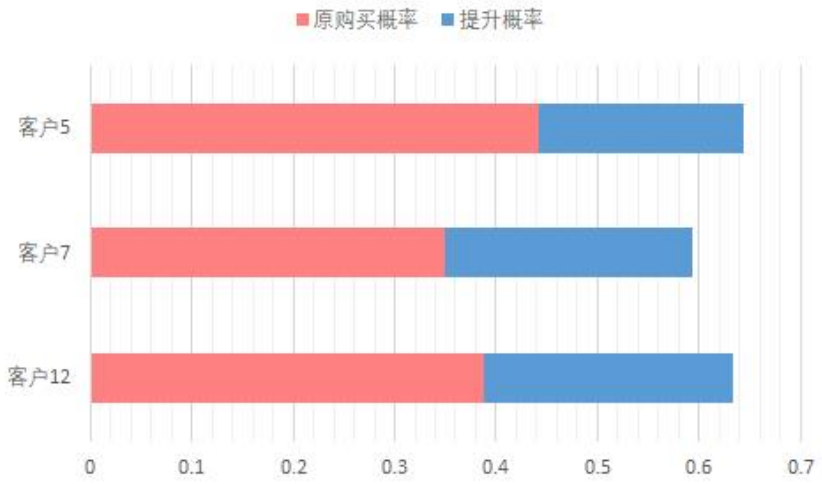

图 15　目标客户购买概率提升图

通过上图可以发现，客户 5、客户 7 和客户 12 对应购买电动汽车的概率都具有显著的提升，并且都大于 50%，通过了客户筛选模型。同时所求解出的客户满意打分各项指标的提升比例趋近于 2%-3%。以上结论说明了构建模型的优越性与合理性，

八、 问题五的模型建立与求解

8.1 问题五的描述与分析

问题五要求根据前四问的研究结论，对销售部门提出营销策略的建议。

8.2 提出建议

结合问题一的研究结果，我们发现客户更倾向于选择自主品牌的电动汽车，这是因为自主品牌为最早进入电动汽车产业的品牌类型，而合资品牌与新势力品牌均为新兴品牌。因此，销售部门在进行营销时，可以针对不同品牌选择不同的人群进行宣传。对于新事物接受程度更大的年轻人群，加大合资品牌与新势力品牌的宣传力度 [7]。

结合问题二的研究结果，我们发现对于不同品牌的电动汽车，客户选择购买时所关注的重要指标也不同。销售部门可以针对不同品牌电动汽车的重要指标进行宣传并扬长避短。

结合问题三的研究结果，销售部门可以对目标客户群体的购买可能性进行预测。因此，在实际进行销售时，可以提前对市场进行调研，从而确定更具有购买意愿的消费群体，进行有针对性的宣传。

结合问题四的研究结果，运用问题四中的客户筛选模型，针对附件一中的客户群体进行计算，得到三种品牌潜在客户数量分别为**30**、**67**、**11**，可以有针对性地是实施营销策略。

从长远发展的角度来看，销售部门的员工需要提升各个方面的素养来提升客户的服务满意度，同时可以了解客户对销售员工的服务需求，更有针对性的提升服务质量，提高销售水平。此外，还需要与市场研发部门进行合作与交流，明确不同品牌电动汽车的研发优势，从而进行更好的宣传。

九、 模型评价

9.1 模型总结

本文研究电动汽车销售的影响因素和营销策略，通过建立灰色关联度评价模型以分析电动汽车综合实力，运用 Local Cascade Ensemble (LCE) 算法预测目标客户购买情况，建立收益-风险规划模型得到有针对性的营销策略。

9.2 模型优缺点分析

优点：

（1）在第一问进行异常值处理过程中，利用主成分分析和 K-means++ 对数据进行分组处理，体现了模型的准确性和逻辑严密性；

（2）在对评价指标赋值时，利用改进的熵权法使评价指标的权重更合理；

（3）本文在问题三中引入了新型算法 Local Cascade Ensemble 算法，提高了模型的求解精度；

（4）本文在问题四中建立了客户挖掘模型对目标客户进行初步筛选，为后续的规划模型建立提供了良好的基础。

缺点：

（1）在问题二中建立评价模型时，未对所选评价指标进行相关性检验。可能部分指标之间相关性较强，会有信息重复表达，对评价结果造成影响；

（2）本文所选的重要指标在 LCE 算法的运用中可能存在模型预测准确率不是最优的问题。

9.3 模型改进与展望

本文研究电动汽车销售的影响因素和营销策略，通过建立灰色关联度评价模型以分析电动汽车综合实力，可以对评价指标进行进一步的处理，降低指标之间的相关性。本文建立收益-风险规划模型得到有针对性的营销策略，可以对约束条件进行进一步改进和优化。

参考文献

[1] 经济分析中概率与统计的应用价值与实践探究. *[J]*. 投资与创业, 32, 2021.

[2] Kevin Fauvel, Élisa Fromont, Véronique Masson, Philippe Faverdin, and Alexandre Termier. Xem: An explainable-by-design ensemble method for multivariate time series classification. *Data Mining and Knowledge Discovery*, 36:1–41, 05 2022.

[3] Mudasir Ganaie, M. Tanveer, Ponnuthurai Suganthan, and Vaclav Snasel. Ensembles of double random forest. 11 2021.

[4] Yuan Ke, Heng Lian, and Wenyang Zhang. High-dimensional dynamic covariance matrices with homogeneous structure. *Journal of Business and Economic Statistics*, 40(1), 2022.

[5] Wan’an Liu, Hong Fan, and Min Xia. Multi-grained and multi-layered gradient boosting decision tree for credit scoring. *Applied Intelligence*, 52, 03 2022.

[6] Hao Zhang and Guixin Cai. Subsidy strategy on new-energy vehicle based on incomplete information: A case in china. *Physica A: Statistical Mechanics and its Applications*, 541:123370, 2020.

[7] 宋瑛琪. 一汽-大众公司 id 车型市场营销策略研究. *[D]*. 吉林大学, 2022.

[8] 张鹏，党世力，黄梅雨. 基于机器学习预测股票收益率的两步骤 m-sv 投资. 组合优化 *[J/OL]*. 中国管理科学, 2021.

[9] 杨永利，胡晓峰，荣明，殷小静，王文祥. 基于机器学习的作战体系能力特征指标挖掘. *[J]*. 系统仿真学报, 31(06):1048–1054, 2019.

[10] 邹志文. 商业银行边际成本对贷款行为的影响研究. *[J]*. 中国市场, (03):43–44, 2022.

[11] 高文璇, 张帆. 基于 qfd 的场景化新能源汽车满意度提升研究. *[J]*. 中国汽车, (06):21–27, 2022.

[12] 黄瑜珈，范思思. 电动汽车与电力营销的影响分析. *[J]*. 电子技术, 40(12):116–117, 2020.

附录 A　支撑材料清单

- **图表**

  -附录 B 各指标描述性统计量数据

- **代码**

  -附录 C 问题一 matlab 源代码

  -附录 D 问题一 python 源代码

  -附录 E 问题二 python 源代码

  -附录 F 问题三 python 源代码

  -附录 G 问题四 python 源代码

附录 B　各指标描述性统计量数据

	标准误差	中位数	方差	峰度	最小值	最大值	求和
a1	0.200706	77.80869	79.116	1.366412	33.15991	99.03689	153049.5
a2	0.195371	77.80347	74.96568	0.302209	51.04517	99.03017	153754.5
a3	0.2359	77.80532	109.2945	0.951054	29.588	99.03255	149067.2
a4	0.196996	77.76704	76.21804	0.289193	51.80685	99.98334	155143.2
a5	0.212183	77.7625	88.42227	1.259387	33.3225	99.9775	151588.6
a6	0.203158	77.77367	81.06011	0.28591	50.26353	99.99187	153241.6
a7	0.197576	77.77433	76.66724	0.315089	51.11287	99.99271	153560.2
a8	0.207312	77.76473	84.40912	0.398095	49.96528	99.98036	152684
B1	0.010471	2	0.215356	-1.19325	1	3	3333
B2	0.258239	20	130.9742	-1.17569	1	60	41997
B3	0.016874	1	0.559186	3.078121	1	6	3160
B4	0.093147	7	17.04046	1.395267	1	30	14998
B5	0.093147	7	17.04046	1.395267	1	30	14998
B6	0.024388	3	1.168184	-0.04528	1	6	6787
B7	0.031981	5	2.008745	0.021756	1	8	8898
B8	0.013634	1	0.365069	-0.00618	0	3	1708
B9	0.112932	1987	25.04816	1.083883	1960	2000	3901871
B10	0.019227	6	0.726015	1.389739	3	8	10742
B11	0.110714	10	24.07385	1.330954	1	38	19800
B12	0.035004	4	2.40642	0.030382	1	9	8677
B13	0.057002	4	6.381392	-0.94623	1	11	9580

附录 C　问题一：matlab 源代码

```
%%问题一
%%数据统计性描述
clc;clear
load('aafter.mat')
a1=zeros(1,9);
a2=zeros(1,9);
a3=zeros(1,9);
c=[0 0 0];
c1=1;
c2=1;
c3=1;
for i=1:1964
    c=[0 0 0];
    for j=1:9
    if a(i,1)==1
        a1(c1,j)=a(i,j);
        c(1)=1;
    end
    if a(i,1)==2
        a2(c2,j)=a(i,j);
    c(2)=1;
    end

    if a(i,1)==3

        a3(c3,j)=a(i,j);
          c(3)=1;
    end
    end
    if c(1)==1
       c1=c1+1;
    end
    if c(2)==1
       c2=c2+1;
    end
    if c(3)==1
       c3=c3+1;
    end

end

av(1,:)=mean(a1)
av(2,:)=mean(a2)%平均值
```

```
av(3,:)=mean(a3)
st(1,:)=std(a1,1)
st(2,:)=std(a2,1)%标准差
st(3,:)=std(a3,1)
for i=1:3
stx(i,:)=st(i,:)./av(i,:);
end

sk(1,:)=skewness(a1)%偏度
sk(2,:)=skewness(a2)
sk(3,:)=skewness(a3)

av=av(:,2:9)
stx=stx(:,2:9)
sk=sk(:,2:9)

%% 灰色关联分析用于综合评价模型例题的讲解
clear;clc
load('av.mat')
load('stx.mat')
load('sk.mat')

X=[
0.582411553 0.373152581 0.399072486
0.577707689 0.859028482 0.601023877
0.571875541 0.053390842 0.574158633]



%% 判断是否需要正向化
[n,m] = size(X);
disp(['共有' num2str(n) '个评价对象, ' num2str(m) '个评价指标'])


Position = input('输入位置: '); %[2,3,4]

    Type = input('输入位置: ') %极小值正向化处理
    % 注意, Position和Type是两个同维度的行向量
    for i = 1 : size(Position,2)
        %这里需要对这些列分别处理, 因此我们需要知道一共要处理的次数, 即循环的次数
       X(:,Position(i)) = Positivization(X(:,Position(i)),Type(i),Position(i));

    end
    disp('正向化后的矩阵 X = ')
    disp(X)
```

```
%% 对正向化后的矩阵进行预处理
z = X ./ repmat(sum(X.*X) .^ 0.5, n, 1);
disp('标准化矩阵 Z = ')
disp(z)


%% 构造母序列和子序列
Y = max(z,[],2); % 母序列为虚拟的，用每一行的最大值构成的列向量表示母序列
X = z; % 子序列就是预处理后的数据矩阵

%% 计算得分
absX0_Xi = abs(X - repmat(Y,1,size(X,2))) % 计算|X0-Xi|矩阵
a = min(min(absX0_Xi)) % 计算两级最小差a
b = max(max(absX0_Xi)) % 计算两级最大差b
rho = 0.5; % 分辨系数取0.5
gamma = (a+rho*b) ./ (absX0_Xi + rho*b) % 计算子序列中各个指标与母序列的关联系数

Z=X;
       if sum(sum(Z<0)) >0 % 如果之前标准化后的Z矩阵中存在负数，则重新对X进行标准化
          disp('原来标准化得到的Z矩阵中存在负数，所以需要对X重新标准化')
          for i = 1:n
             for j = 1:m
                Z(i,j) = [X(i,j) - min(X(:,j))] / [max(X(:,j)) - min(X(:,j))];
             end
          end
          disp('X重新进行标准化得到的标准化矩阵Z为：')
          disp(Z)
       end
       weight = Entropy_Method(Z);
       disp('熵权法确定的权重为：')
       disp(weight)



score = sum(X .* repmat(weight,size(X,1),1),2); % 未归一化的得分
stand_S = score / sum(score); % 归一化后的得分
[sorted_S,index] = sort(stand_S ,'descend') % 进行排序




function [posit_x] = Positivization(x,type,i)

   if type == 1 %极小型
```

```
        disp(['第' num2str(i) '列是极小型，正在正向化'] )
        posit_x = Min2Max(x); %调用Min2Max函数来正向化
        disp(['第' num2str(i) '列极小型正向化处理完成'] )
        disp('~~~~~~~~~~~~~~~~~~~~分界线~~~~~~~~~~~~~~~~~~~~')
    elseif type == 2 %中间型
        disp(['第' num2str(i) '列是中间型'] )
        best = input('请输入最佳的那一个值：  ');
        posit_x = Mid2Max(x,best);
        disp(['第' num2str(i) '列中间型正向化处理完成'] )
        disp('~~~~~~~~~~~~~~~~~~~~分界线~~~~~~~~~~~~~~~~~~~~')
    elseif type == 3 %区间型
        disp(['第' num2str(i) '列是区间型'] )
        a = input('请输入区间的下界：  ');
        b = input('请输入区间的上界：  ');
        posit_x = Inter2Max(x,a,b);
        disp(['第' num2str(i) '列区间型正向化处理完成'] )
        disp('~~~~~~~~~~~~~~~~~~~~分界线~~~~~~~~~~~~~~~~~~~~')
    else
        disp('没有这种类型的指标，请检查Type向量中是否有除了1、2、3之外的其他值')
    end
end

%正向化
function [posit_x] = Min2Max(x)
%  y=input('1还是2：  ')
%  if y==1
%     posit_x = max(x)-x;%正向化处理
%  end
%  if y==2
%     posit_x = max(x)./x;%正向化处理
 posit_x = (max(x)-x)./(max(x)-min(x))

end

% 重新定义一个mylog函数，当输入的p中元素为0时，返回0
function [lnp] = mylog(p)
n = length(p); % 向量的长度
lnp = zeros(n,1); % 初始化最后的结果
    for i = 1:n % 开始循环
        if p(i) == 0 % 如果第i个元素为0
            lnp(i) = 0; % 那么返回的第i个结果也为0
        else
            lnp(i) = log(p(i));
        end
    end
end
```

```
%改进熵权法
function [W] = Entropy_Method(Z)
% 计算有n个样本，m个指标的样本所对应的的熵权
% 输入
% Z ： n*m的矩阵（要经过正向化和标准化处理，且元素中不存在负数）
% 输出
% W: 熵权，1*m的行向量

%% 计算熵权
alpha=3
    [n,m] = size(Z);
    D = zeros(1,m); % 初始化保存信息效用值的行向量
    for i = 1:m
        x = Z(:,i); % 取出第i列的指标
        p = x / sum(x);
        % 注意，p有可能为0，此时计算ln(p)*p时，Matlab会返回NaN，所以这里我们自己定义一个函数
        e(i) = -sum(p .* mylog(p)) / log(n); % 计算信息熵
        D(i) = 1- e(i); % 计算信息效用值
    end
    W = D ./ sum(D); % 将信息效用值归一化，得到权重
    %%权重改进
    ae=mean(e)
    w=mean(W);
    for i=1:m
        if W(i)~=0
    W(i)=(1-ae^alpha)*W(i)+ae^alpha*w;

        else
            W(i)=0;
        end
    end
end
```

附录 D　问题一：python 源代码

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import scipy.stats as stats
import statsmodels.api as sm
from scipy.stats import chi2_contingency
import math

pension = pd.read_excel("养老保险.xlsx")
```

```
pension

sum = pension[["基本养老保险基金总收入","年份"]]
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', rc=rc)

plt.figure(dpi=1000) # 设置分辨率
sns.lmplot(x='年份',y='基本养老保险基金总收入',data=sum)

sum.columns=["money","year"]
fit=sm.formula.ols('money~year',data=sum).fit()
print(fit.params)
preX = pd.Series([2021,2022,2023,2024,2025],name='year')
predicty = fit.predict(preX)
predicty

print(fit.summary())


sum1 = pension[["基金总支出","年份"]]
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', rc=rc)

plt.figure(dpi=1000) # 设置分辨率
sns.lmplot(x='年份',y='基金总支出',data=sum1)

sum1.columns=["money","year"]
fit=sm.formula.ols('money~year',data=sum1).fit()
print(fit.params)
predicty = fit.predict(preX)
predicty

print(fit.summary())

sum2 = pension[["年末基本养老保险基金累计结存","年份"]]
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', rc=rc)

plt.figure(dpi=2000) # 设置分辨率
sns.lmplot(x='年份',y='年末基本养老保险基金累计结存',data=sum2)

sum2.columns=["money","year"]
fit=sm.formula.ols('money~year',data=sum2).fit()
print(fit.params)
```

```
predicty = fit.predict(preX)
predicty

print(fit.summary())

sum3 = pension[["企业年金","年份"]].copy()
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
sns.set(context='notebook', rc=rc)

plt.figure(dpi=1000) # 设置分辨率
sns.lmplot(x='年份',y='企业年金',data=sum3)

sum3["企业年金"] = sum3["企业年金"].apply(np.log1p)
sum3

sum3.columns=["money","year"]
sum3["year2"] = sum3["year"]**2
sum3

fit = sm.OLS(sum3["money"],sum3[["year","year2"]]).fit()
print(fit.params)
preXFor = pd.DataFrame({'year':preX.values,'year2':preX.values**2})

predicty = fit.predict(preXFor)

predicty

print(fit.summary)

np.expm1(predicty)

print(fit.summary())

def checkVIF(df):
    from statsmodels.stats.outliers_influence import variance_inflation_factor
    name = df.columns
    x = np.matrix(df)
    VIF_list = [variance_inflation_factor(x,i) for i in range(x.shape[1])]
    VIF = pd.DataFrame({'feature':name,"VIF":VIF_list})
    max_VIF = max(VIF_list)
    print(max_VIF)
    return VIF

lasso = pd.read_excel("lasso.xlsx",sheet_name="Lasso")
lasso = lasso.drop(["年数"],axis=1)
checkVIF(lasso)
```

```
checkVIF(lasso).to_excel("lasso-vif.xlsx")

spear = pd.read_excel("lasso.xlsx",sheet_name="spearman",index_col=0)
spear.head()

col_name = spear.columns.values
sns.heatmap(spear,annot=True,xticklabels=col_name,yticklabels=col_name,cmap='GnBu')

spear1 = pd.read_excel("pearson.xlsx",index_col=0)
spear1.head()

spear1.rename(columns={"参加养老保险人数(万人)":"参加养老保险人数"},inplace=True)

col_name = spear1.columns.values
sns.heatmap(spear1,annot=True,xticklabels=col_name,yticklabels=col_name,cmap='GnBu')
```

附录 E　问题二：python 源代码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure, show, rc
import seaborn as sns
rc = {'font.sans-serif': 'SimHei',
      'axes.unicode_minus': False}
# 各个省份城乡人口比
proRate = pd.read_excel("各省份城乡人口.xls",index_col=0)
proRate.head(2)
# 各省份老年人口数
oldNum = pd.read_excel("各省份老年人口.xlsx",index_col=0)
oldNum.head(2)
# 拟合后真实的服务床数增加
realBed = 1969.910683
# 需求率
needNate = 0.1
res = pd.concat([proRate,oldNum],axis=1)
res.head()
res["城乡人口比"] = res["城镇人口（万）"]*1.0/res["乡村人口（万）"]
res["服务床位数"] = realBed*res["2025真实人口"]/res["2025真实人口"].sum()
res["缺口"] = res["老年人（万人）"] * needNate - res["服务床位数"]
res["待发掘"] = res["老年人（万人）"]*(1-needNate)
endRes = res[["缺口","待发掘","城乡人口比"]]
endRes.to_excel("forGrey.xlsx")
```

```
# 绘图
mydata = pd.DataFrame(dict(year=["2021年", "2022年", "2023年", "2024年", "2025年"],
                           money=[13874.33333, 16351.85433, 19241.82333, 22552.72333,
                               26293.03833]))
n_row = mydata.shape[0]
angle = np.arange(0, 2*np.pi, 2*np.pi/n_row)
radius = np.array(mydata.money)

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
fig = figure(figsize=(4, 4), dpi=1000)
# 极坐标条形图，polar为True
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8], polar=True)

# 方法用于设置角度偏离,参数值为弧度值数值
ax.set_theta_offset(np.pi/2-np.pi/n_row)
#
    当set_theta_direction的参数值为1,'counterclockwise'或者是'anticlockwise'的时候，正方向为逆时针;
# 当set_theta_direction的参数值为-1或者是'clockwise'的时候，正方向为顺时针;
ax.set_theta_direction(-1)
# 方法用于设置极径标签显示位置,参数为标签所要显示在的角度
ax.set_rlabel_position(360-180/n_row)

plt.bar(angle, radius, color='#70A6FF', edgecolor="k", width=1.245, alpha=0.8)
# plt.bar(angle,radius, color='#F9837A',edgecolor="k",width=0.90,alpha=0.9)
# x轴坐标轴标签
plt.xticks(angle, labels=mydata.year, size=12)
plt.ylim(-15, 125)
# plt.ylim(0,125)
plt.yticks(np.arange(0, 35000, 5000), verticalalignment='center',
         horizontalalignment='right')

plt.grid(which='major', axis="x", linestyle='-',
        linewidth='0.5', color='gray', alpha=0.5)
plt.grid(which='major', axis="y", linestyle='-',
        linewidth='0.5', color='gray', alpha=0.5)

plt.show()
```

附录 F　问题三：python 源代码

```
import BaseICA
import numpy as np
def fitFunc(solution):
   # 第一个约束
   def g1(x):
```

```
        return (2-4*x[0])*81*x[4] + (2-4*x[1])*220*x[5] + \
            (2-4*x[2])*397*x[6] + (2-4*x[3])*924.5*x[7] - 2346753

    def g2(x):
      return 15000 - (x[4]+x[5]+x[6]+x[7])

    def violate(value):
        return -1e8 if value <=0 else 0

    w = 1

    fx = 42.7*solution[0]*solution[4]+109.8*solution[1]*solution[5] + \
        201.7*solution[2]*solution[6]+435.5*solution[3]*solution[7] - \
        16.8*solution[4]-34.1*solution[5]-58.63*solution[6]-121.8*solution[7] + \
        w * (14.3*solution[4]+18.8*solution[5]+24.7*solution[6]+42*solution[7])

    fx += violate(g1(solution)) + violate(g2(solution))

    return fx

problem = {
    "fit_func":fitFunc,
    "lb":[0,0,0,0,0,0,0,0], #lower
    "ub":[0.5,0.5,0.5,0.5,15000,15000,15000,15000], #upper
    "minmax":"max"
}

epoch = 1000
pop_size = 200
empire_count = 20
assimilation_coeff = 1.5
revolution_prob = 0.05
revolution_rate = 0.1
revolution_step_size = 0.1
zeta = 0.1

model =
     BaseICA(problem,epoch,pop_size,empire_count,assimilation_coeff,revolution_prob,revolution_rate,revolution_s

best_position, best_fitness = model.solve()
print(f"Solution: {best_position}, Fitness: {best_fitness}")
model.history.save_global_best_fitness_chart(filename="problem2Res/gbfc")


# 帝国竞争算法
import numpy as np
from copy import deepcopy
```

```
class BaseICA():
    def __init__(self, problem, epoch=10000, pop_size=100, empire_count=5,
        assimilation_coeff=1.5,
               revolution_prob=0.05, revolution_rate=0.1, revolution_step_size=0.1, zeta=0.1,
                   **kwargs):
        super().__init__(problem, kwargs)
        self.epoch = self.validator.check_int("epoch", epoch, [1, 100000])
        self.pop_size = self.validator.check_int("pop_size", pop_size, [10, 10000])
        self.empire_count = self.validator.check_int("empire_count", empire_count, [2, 2 +
            int(self.pop_size / 5)])
        self.assimilation_coeff = self.validator.check_float("assimilation_coeff",
            assimilation_coeff, [1.0, 3.0])
        self.revolution_prob = self.validator.check_float("revolution_prob", revolution_prob,
            (0, 1.0))
        self.revolution_rate = self.validator.check_float("revolution_rate", revolution_rate,
            (0, 1.0))
        self.revolution_step_size = self.validator.check_float("revolution_step_size",
            revolution_step_size, (0, 1.0))
        self.zeta = self.validator.check_float("zeta", zeta, (0, 1.0))

        self.nfe_per_epoch = self.pop_size
        self.sort_flag = True
        self.pop_empires, self.pop_colonies, self.empires = None, None, None
        self.n_revoluted_variables, self.idx_list_variables = None, None

    def revolution_country__(self, position, idx_list_variables, n_revoluted):
        pos_new = position + self.revolution_step_size * np.random.normal(0, 1,
            self.problem.n_dims)
        idx_list = np.random.choice(idx_list_variables, n_revoluted, replace=False)
        position[idx_list] = pos_new[idx_list]
        return position

    def after_initialization(self):
        self.pop, self.g_best = self.get_global_best_solution(self.pop)
        self.n_revoluted_variables = int(round(self.revolution_rate * self.problem.n_dims))
        self.idx_list_variables = list(range(0, self.problem.n_dims))

        colony_count = self.pop_size - self.empire_count
        self.pop_empires = deepcopy(self.pop[:self.empire_count])
        self.pop_colonies = deepcopy(self.pop[self.empire_count:])

        cost_empires_list = np.array([solution[self.ID_TAR][self.ID_FIT] for solution in
            self.pop_empires])
        cost_empires_list_normalized = cost_empires_list - (np.max(cost_empires_list) +
            np.min(cost_empires_list))
        prob_empires_list = np.abs(cost_empires_list_normalized /
            np.sum(cost_empires_list_normalized))
```

```
        self.empires = {}
        idx_already_selected = []
        for i in range(0, self.empire_count - 1):
            self.empires[i] = []
            n_colonies = int(round(prob_empires_list[i] * colony_count))
            idx_list = np.random.choice(list(set(range(0, colony_count)) -
                set(idx_already_selected)), n_colonies, replace=False).tolist()
            idx_already_selected += idx_list
            for idx in idx_list:
                self.empires[i].append(self.pop_colonies[idx])
        idx_last = list(set(range(0, colony_count)) - set(idx_already_selected))
        self.empires[self.empire_count - 1] = []
        for idx in idx_last:
            self.empires[self.empire_count - 1].append(self.pop_colonies[idx])

    def evolve(self, epoch):
        nfe_epoch = 0
        for idx, colonies in self.empires.items():
            for idx_colony, colony in enumerate(colonies):
                pos_new = colony[self.ID_POS] + self.assimilation_coeff * \
                        np.random.uniform(0, 1, self.problem.n_dims) *
                            (self.pop_empires[idx][self.ID_POS] - colony[self.ID_POS])
                pos_new = self.amend_position(pos_new, self.problem.lb, self.problem.ub)
                self.empires[idx][idx_colony][self.ID_POS] = pos_new
                if self.mode not in self.AVAILABLE_MODES:
                    self.empires[idx][idx_colony][self.ID_TAR] = self.get_target_wrapper(pos_new)
            self.empires[idx] = self.update_target_wrapper_population(self.empires[idx])
            nfe_epoch += len(self.empires[idx])

        for idx, colonies in self.empires.items():
            pos_new_em = self.revolution_country__(self.pop_empires[idx][self.ID_POS],
                self.idx_list_variables, self.n_revoluted_variables)
            pos_new_em = self.amend_position(pos_new_em, self.problem.lb, self.problem.ub)
            self.pop_empires[idx][self.ID_POS] = pos_new_em
            if self.mode not in self.AVAILABLE_MODES:
                self.pop_empires[idx][self.ID_TAR] = self.get_target_wrapper(pos_new_em)

            for idx_colony, colony in enumerate(colonies):
                if np.random.rand() < self.revolution_prob:
                    pos_new = self.revolution_country__(colony[self.ID_POS],
                        self.idx_list_variables, self.n_revoluted_variables)
                    pos_new = self.amend_position(pos_new, self.problem.lb, self.problem.ub)
                    self.empires[idx][idx_colony][self.ID_POS] = pos_new
                    if self.mode not in self.AVAILABLE_MODES:
                        self.empires[idx][idx_colony][self.ID_TAR] =
                            self.get_target_wrapper(pos_new)
```

```
            self.empires[idx] = self.update_target_wrapper_population(self.empires[idx])
            nfe_epoch+= len(self.empires[idx])
        self.pop_empires = self.update_target_wrapper_population(self.pop_empires)
        self.update_global_best_solution(self.pop_empires, save=False)
        nfe_epoch += len(self.pop_empires)

        for idx, colonies in self.empires.items():
            for idx_colony, colony in enumerate(colonies):
                if self.compare_agent(colony, self.pop_empires[idx]):
                    self.empires[idx][idx_colony], self.pop_empires[idx] =
                        deepcopy(self.pop_empires[idx]), deepcopy(colony)

        cost_empires_list = []
        for idx, colonies in self.empires.items():
            fit_list = np.array([solution[self.ID_TAR][self.ID_FIT] for solution in colonies])
            fit_empire = self.pop_empires[idx][self.ID_TAR][self.ID_FIT] + self.zeta *
                np.mean(fit_list)
            cost_empires_list.append(fit_empire)
        cost_empires_list = np.array(cost_empires_list)

        cost_empires_list_normalized = cost_empires_list - (np.max(cost_empires_list) +
            np.min(cost_empires_list))
        prob_empires_list = np.abs(cost_empires_list_normalized /
            np.sum(cost_empires_list_normalized))

        uniform_list = np.random.uniform(0, 1, len(prob_empires_list))
        vector_D = prob_empires_list - uniform_list
        idx_empire = np.argmax(vector_D)

        idx_weakest_empire = np.argmax(cost_empires_list)
        if len(self.empires[idx_weakest_empire]) > 0:
            colonies_sorted, best, worst =
                self.get_special_solutions(self.empires[idx_weakest_empire])
            self.empires[idx_empire].append(colonies_sorted.pop(-1))
        else:
            self.empires[idx_empire].append(self.pop_empires.pop(idx_weakest_empire))

        self.pop = self.pop_empires + self.pop_colonies
```