所属类别	2025 年“华数杯”全国大学生数学建模竞赛	参赛编号
本科组		CM******

标题（此处换成论文的标题）

摘　要

随着 xx 的发展,xx 问题是 xx 中的重要研究课题（或 xx 现象日益严重）。本文针对 xx 中的问题，基于 xx 和 xx 思想, 通过确定 xx、xx、xx 等指标，以 xx、xx 为目标建立了 xx 模型，并使用 xx 算法对模型进行求解。

针对问题一，

针对问题二，

针对问题三，

最后, 我们对提出的模型进行全面的评价：本文的模型贴合实际, 能合理解决提出的问题, 具有实用性强, 算法效率高等特点, 该模型在 xx,xx,xx 方面也能使用。

欢迎关注我们的微信公众号：

关键词：关键词 1 关键词 2 关键词 3 关键词 4 关键词 5

一、问题的提出

1.1 问题的背景

随着……的发展，

1.2 问题的提出

(1) 问题一：
(2) 问题二：
(3) 问题三：
(4) 问题四：
(5) 问题五：

二、问题的分析

2.1 问题一的分析

……

2.2 问题二的分析

……

2.3 问题三的分析

……

2.4 问题四的分析

……

三、模型的假设

由于……，所以也需要遵循一定的假设

假设 1……
假设 2……
假设 3……
假设 4……
假设 5……
假设 6……

假设 7……

四、符号说明

表 1 变量符号及其解释说明

符号	解释说明
a_i	第 i 个满意度
B_i	第 i 个目标客户体验者信息
Q_i	变量的第 i 分位数
IQR	四分位距
a_{mean}	a_1~a_8 的平均值

五、模型的建立与求解

5.1 问题一

5.1.1 缺失值的处理

经初步检查，本题数据中的缺失值只有附录一中 B_7 变量下出现的 NULL 值。根据附录二中的问卷信息，B_7 的含义是“有几个小孩”，批量出现 NULL 值的原因只能是用户没有填写，即“没有小孩”，故在 SPSS 中将“NULL”替换为“0”

5.1.2 异常值的处理

异常值的定义：将不符合附件二中问卷填写要求的变量值以及符合问卷填写要求但明显离群的标度值定义为本题的异常值。

异常值的处理原理：首先，由于数据来源于目标客户体验者填写的问卷，应选取填写规范的问卷信息作为有效数据。对于分类变量的变量值，不能出现问卷中没有定义的类别；对于数值变量的变量值，不能超出问卷中所规定的数值范围。

其次，对于某些数值变量，存在一些离群度较大的离群点，也应该认定为异常值。由于满意度（a_1~a_8）受目标客户体验者主观因素以及各种外界因素的影响大，容易出现歧异值，应该清除离群度较高的值；而对于目标客户体验者信息的数据（B_1~B_{17}），由于客户样本的多样性及客户信息的客观性，去除离群值可能对模型的适用性产生影响，因此在本题中不对其离群值做处理。

对于离群值的选择，我们将箱线图中距离箱体边缘（上四分位点、下四分位点）的距离大于箱体的 3 倍的值定义为离群点。以品牌 1 的满意度箱线图为例，如图 1，用

* 标出的数据由于过大或过小而超出正常值范围，是离群点，应该清除。

图 1 箱线图表示离群值

由于品牌 1 到品牌 3 的目标客户体验者是不同的群体，在清除离群值的时候分组进行。

异常值的处理步骤：

(1) 过滤不合规范的数据：根据附件二确定出 a_1~a_8、B_1~B_{17} 的取值范围：

$$\begin{cases} 0 \leqslant a_i \leqslant 100, & i = 1,2,3...8 \\ 1 \leqslant B_1 \leqslant 3, & B_1 \in Z \\ 1 \leqslant B_3 \leqslant 6, & B_3 \in Z \\ 1 \leqslant B_6 \leqslant 8, & B_6 \in Z \\ 1 \leqslant B_{11} \leqslant 9, & B_{11} \in Z \\ 1 \leqslant B_{12} \leqslant 11, & B_{12} \in Z \end{cases}$$

以该取值范围为筛选条件，在 SPSS 中选择符合条件的个案

(2) 过滤离群值：按照上文对离群值的定义，确定出非离群值的筛选条件：

$$\begin{cases} \dfrac{x - Q_3}{IQR} \leqslant 3 \\ \dfrac{Q_1 - x}{IQR} \leqslant 3 \end{cases}$$

其中，x 为待清洗的变量值，Q_3 为第三分位数，Q_1 为第一分位数，IQR 为四分位距。以该筛选条件为依据，在 SPSS 中选出符合要求的个案。

异常值的处理效果评价：对处理前后的数据分别计算方差，得到如表 2、表 3、表 4中的数据：

表 2 品牌 1 数据处理前后各变量值的方差比较

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
处理前	87.94	82.79	110.16	86.04	90.47	90.48	89.40	101.22
处理后	79.42	75.66	102.83	77.63	83.87	83.51	82.73	94.14

表 3 品牌 2 数据处理前后各变量值的方差比较

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
处理前	74.05	80.74	108.48	80.64	88.65	85.30	82.89	85.94
处理后	67.58	74.59	100.61	72.62	79.86	78.64	72.54	78.21

将该表格制作成折线图，如图 2所示，可以看到对于三个品牌，a_1~a_8 在处理后方差均有明显的减小，异常值处理效果较好。

表 4 品牌 3 数据处理前后各变量值的方差比较

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
处理前	92.72	91.49	115.63	102.03	93.34	100.52	107.65	91.77
处理后	90.13	81.15	101.48	98.55	87.84	93.03	94.06	79.21

(a) 品牌 1 (b) 品牌 2

(c) 品牌 3

图 2 异常值处理前后方差比较

5.1.3 数据的描述及可视化

(1) 各品牌目标客户体验者人数及有购买意愿的人数占比分析：

表 5 各品牌目标客户体验者人数及有购买意愿的人数占比

	有购买意愿的人数	无购买意愿的人数	总人数	有购买意愿人数占比
品牌 1	23	531	554	4.2%
品牌 2	65	1197	1262	5.2%
品牌 3	11	124	135	8.1%
总计	99	1852	1951	5.1%

如表 5所示，品牌 2 的目标客户体验者人数最多，而品牌 3 的目标客户体验者人数最少; 三个品牌有购买意愿的体验者人数占比都较低，低于 10%，其中品牌 3 最高，有 8.1%, 品牌 2 次之，品牌 1 最低。可视化结果如图 3所示。

(2) 目标客户对不同品牌的满意度比较分析：

对 a_1~a_8 取平均值生成变量 a_{mean}。按品牌对 a_{mean} 进行可视化如图 4。

图 3 各品牌客户人数及购买意愿

图 4 各品牌满意度平均值比较

并在 SPSS 中对 a_mean 进行单因素 ANOVA 分析，分析结果如图 5

图 5 a_{mean}ANOVA 分析

令显著性水平为 0.05，由分析结果可知，$F = 1.782$、$p = 0.169$，小于显著性水平，由此可见客户对三个品牌电动车的满意度平均值差别不大。

(3) 有意愿购买人群和没有意愿购买人群的满意度比较：

按照是否有购买意愿将数据分组，分别对两组数据中的 a_1~a_8 取平均值如表 6、表 7：

表 6 愿意购买的人群各满意度平均值

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
品牌 1	86.04	85.98	85.53	86.03	84.22	85.21	84.80	84.78
品牌 2	87.33	86.20	85.82	86.90	85.87	85.21	85.95	84.74
品牌 3	89.11	89.56	86.44	86.42	86.92	86.72	84.83	86.28

表 7 不愿意购买的人群各满意度平均值

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
品牌 1	77.79	78.51	75.77	79.28	77.75	78.08	78.50	78.13
品牌 2	77.63	77.67	75.63	78.41	76.70	77.57	77.56	77.09
品牌 3	76.15	76.38	73.64	77.21	74.96	76.31	77.05	76.56

将平均值制作成多线折线图 (图 6、图 7)。可以看到，有购买意愿的人群对电动车的满意度比没有购买意愿的人群整体要高许多；没有购买意愿的人对电动车各方面的满意度呈现出相同的趋势，即对经济性 (a_3) 满意度较低，对舒适性 (a_2) 和安全性 (a_4) 满意度较高。

图 6 愿意购买的人群满意度平均值

图 7 不愿意购买的人群满意度平均值

(4) 有意愿购买人群和没有意愿购买人群的经济状况比较：

表 8 不同购买意愿人群经济情况比较

a 家庭年收入平均值比较

	无意愿购买	有意愿购买
品牌 1	28.766	31.173
品牌 2	24.929	31.230
品牌 3	32.254	31.727

b 个人年收入平均值比较

	无意愿购买	有意愿购买
品牌 1	17.08	19.87
品牌 2	15.85	19.18
品牌 3	19.38	17.64

5.2 问题二

5.2.1 方法概要

判断哪些因素会影响到购买意愿可以采用回归分析，由于因变量 (购买意愿) 是二分变量，应该采用 Logistic 回归。

如果将 a_1~a_8 、B_1~B_{17} 全部变量直接代入回归方程检验，会因为变量太多导致回归效果欠佳。因此应该首先筛选出在不同购买意愿群体中有统计性差异的变量，再用筛选出的变量进行回归分析，以提高回归效果。

5.2.2 变量筛选

由第一问可知：对于不同购买意愿的群体，用户的满意度（aa_1~a_8）和用户的经济情况有显著差异，因此应该将 a_1~a_8 以及反映用户经济情况的变量 (B_{10}、B_{13}、B_{14}、B_{15}、B_{16}、B_{17}) 纳入回归方程的检验中。其中 B_{10} 代表工作年限、B_{13} 代表家庭年收入、B_{14} 代表个人年收入、B_{15} 家庭可支配年收入、B_{16} 代表房贷支出比例、B_{17} 代表车贷支出比例。

对于其他统计性差异未知的变量，如户口情况、居住区域等，应逐个进行检验。

对于分类变量 (B_1、B_3、B_5、B_6、B_{12}), 采用卡方检验，检验结果汇总如下：

表 9　品牌 1 分类变量卡方检验结果

	Pearson 卡方值	渐进显著性 (双侧)	是否纳入回归方程
B_1	0.253	0.881	否
B_3	27.136	0.000	是
B_5	3.019	0.697	否
B_6	2.796	0.834	否
B_{12}	8.766	0.554	否

表 10　品牌 2 分类变量卡方检验结果

	Pearson 卡方值	渐进显著性 (双侧)	是否纳入回归方程
B_1	1.643	0.440	否
B_3	3.584	0.465	否
B_5	2.830	0.726	否
B_6	3.082	0.687	否
B_{12}	19.338	0.036	是

表 11 品牌 3 分类变量卡方检验结果

	Pearson 卡方值	渐进显著性 (双侧)	是否纳入回归方程
B_1	0.473	0.789	否
B_3	2.685	0.612	否
B_5	4.579	0.469	否
B_6	3.252	0.861	否
B_{12}	6.227	0.712	否

综上，每个品牌中纳入回归分析的变量有：**品牌 1**：a_1~a_8、B_2、B_{10}、B_{13}~B_{17} **品牌 2**：a_1~a_8、B_2、B_{10}、B_{12}~B_{17} **品牌 3**：a_1~a_8、B_{10}、B_{13}~B_{17}

(3) 回归分析：在 SPSS 中分品牌对 (2) 中筛选出的变量进行二元 Logistic 回归分析，选取 0.1 作为显著性，认为在检验结果中 P 值小于 0.1 的变量是可能对购买意愿产生影响的变量，得到的回归结果如表 12

表 12 可能对购买意愿产生影响的变量

品牌	可能对购买意愿造成影响的因素
品牌 1	a_1、a_3、B_{16}、B_{17}
品牌 2	a_1、a_3、B_{15}、B_{16}、B_{17}、a_5、B_{12}
品牌 3	B_{16}、a_2、a_7、a_3、B_{10}

即，对品牌 1 来说，电池技术性能、经济性、房贷支出比例、车贷支出比例可能影响客户的购买意愿；对品牌 2 来说，电池技术性能、经济性、动力性、职位、家庭可支配收入、房贷支出比例、车贷支出比例可能影响客户的购买意愿；对品牌 3 来说，舒适性、经济性、外观内饰、工作年限、房贷支出比例可能影响客户购买意愿。

5.3 问题三

第三题采用了多种方法进行不同品牌电动汽车的客户挖掘模型的建立，主要是 CART 决策树的生成和神经网络模型两种方法

5.3.1 CART 决策树的生成

基本原理：CART 决策树的生成就是递归地构建二叉决策树的过程。结点最优划分特征的选择使用基尼指数（Gini index），其中主要决定各个因素对于客户模型影响的程度，考虑到此处的数据较为分散，对影响的因素根据数据大小进行分组。

先介绍基尼指数的定义：分类问题中，假设有 K 个类，样本点属于第 k 类的概率为 p_k，则概率分布的基尼指数定义为

$$Gini(p) = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2 \tag{1}$$

对于二分类问题，若样本点属于第 1 个类的概率是 p，则概率分布的基尼指数为

$$Gini(p) = 2p(1-p) \tag{2}$$

对于给定的样本集合 D，其基尼指数为

$$Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2 \tag{3}$$

在此处，C_k 是 D 中属于 k 类的样本子集，K 是类的个数。

如果样本集合 D 根据特征 A 是否取某一可能值 a 被分割成 D_1 和 D_2 两部分, 即

$$D_1 = (x, y) \in D | A(x) = a, D_2 = D - D_1 \tag{4}$$

则在特征 A 的条件下，集合 D 的基尼指数定义为

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{5}$$

基尼指数 Gini(D) 表示集合 D 的不确定性，基尼指数 Gini(D, A) 表示经 A=a 分割后集合 D 的不确定性。基尼指数值越大，样本集合的不确定性也就越大，这一点与熵相似。

图 8 二分类中基尼指数、信息熵、和分类误差率的关系

CART 生成算法：

输入: 训练数据集 D，停止计算的条件;

输出:CART 决策树。

根据训练数据集，从根结点开始，递归地对每个结点进行以下操作，构建二叉决策树:

(1）设结点的训练数据集为 D，计算现有特征对该数据集的基尼指数。此时，对每一个特征 A，对其可能取的每个值 a，根据样本点对 A= a 的测试为“是”或“否”将 D 分割成 D_1 和 D_2 部分，利用式(5) 式计算 A= a 时的基尼指数。

(2) 在所有可能的特征 A 以及它们所有可能的切分点 a 中，选择基尼指数最小的特征及其对应的切分点作为最优特征与最优切分点。依最优特征与最优切分点，从现结点生成两个子结点，将训练数据集依特征分配到两个子结点中去。

(3) 对两个子结点递归地调用 (1)，(2)，直至满足停止条件。(4) 生成 CART 决策树。

算法停止计算的条件是结点中的样本个数小于预定阈值，或样本集的基尼指数小于预定阈值 (样本基本属于同一类)，或者没有更多特征。

具体的情况如下文所示：

决策树模型结果： (1) 品牌 1: 与城市居住龄高度相关，但出来决策树效果不好，这里我加上了弱相关的因素（Person 系数大于 0.17）

对于品牌 1，对客户购买意愿的影响因素有：房贷占比，车贷占比，电池，经济

图 9 品牌 1 决策树模型

对模型进行精度检验，得到模型评价精度 94.5，评价结果如图 10

图 10 品牌 1 决策树模型评价精度

(2) 品牌 2: 对于品牌 2，外观，动力，安全，经济，舒适，电池与客户购买意愿有一般正相关关系，与房贷占比有一般负相关关系，与个人年收入和单位有非常紧密关系，与城市居住龄有紧密关系。

图 11 品牌 2 决策树模型

对模型进行精度检验，得到模型评价精度 92.86，评价结果如图 12

图 12 品牌 2 决策树模型评价精度

(3) 品牌 3: 对于品牌 3，客户购买意愿与电池，品质，外观，操控性，动力性，安全性，经济型，舒适性，具有一般线性关系

图 13 品牌 3 决策树模型

对模型进行精度检验，得到模型评价精度 88.89，评价结果如图 13

对附录三中的客户数据进行预测，预测结果为：

图 14 品牌 2 决策树模型评价精度

对于品牌 1：2、4、5 号顾客有购买的意愿，1、3 号顾客没有购买的意愿; 对于品牌 2：7、10 号有购买的意愿，6、8、9 号顾客没有购买的意愿; 对于品牌 3：11、12 号顾客有购买的意愿，13、14、15 号顾客没有购买的意愿

5.3.2 神经网络模型

采用神经网络模型，其中输入的影响的因素采用的问题二中的影响因素，并将其作为输入部分的参数，将最后结果是否有购买意愿的 0,1 作为标签。本处使用卷积神经网络 CNN 是一种前馈神经网络，主要原理步骤如下：

给定一个输入信号序列　和滤波器 , 卷积的输出为：

$$y_t = \sum_{k=1}^{m} w_k x_{t-k+1} \tag{6}$$

本次采用的二维卷积如图 15：

图 15　二维卷积

用卷积层代替全连接层：

图 16　卷积层

框图结构如图 17：

图 17　卷积层

模型的精度和损失函数的情况随着训练次数的情况如图 18所示：

图 18　神经网络模型精度

由图易得随着训练次数的上升，整体的训练精度提高，但是预测精度会稍稍下降，这是因为训练的精度过高后易发生模型的过拟合现象，从曲线上看，模型整体的拟合效果较好。

利用模型进行预测结果如图 19：

图 19　神经网络模型预测结果

对于品牌 1：2、4、5 号顾客有购买的意愿，1、3 号顾客没有购买的意愿; 对于品牌 2：7、10 号有购买的意愿，6、8、9 号顾客没有购买的意愿; 对于品牌 3：11、12 号顾客有购买的意愿，13、14、15 号顾客没有购买的意愿

5.3.3 预测结果

第三题采用了 CART 决策树的生成和神经网络模型两种方法进行挖掘模型的拟合，两种方法的模型的拟合效果和预测效果均比较好，尤其是卷积神经网络网络，训练精度高达 96.67%，可根据训练的次数去决定训练精度和预测精度，最后的 15 名目标客户购买电动车的可能性如下：

对于品牌 1：2、4、5 号顾客有购买的意愿，1、3 号顾客没有购买的意愿; 对于品牌 2：7、10 号有购买的意愿，6、8、9 号顾客没有购买的意愿; 对于品牌 3：11、12 号顾客有购买的意愿，13、14、15 号顾客没有购买的意愿

5.4 问题四

第四问中需要基于这种思路和前面的研究成果，在附件 3 每个品牌中各挑选 1 名没有购买电动汽车的目标客户，实施销售策略。我们根据第三问的结果：**对于品牌 1：2、4、5 号顾客有购买的意愿，1、3 号顾客没有购买的意愿；对于品牌 2：7、10 号有购买的意愿，6、8、9 号顾客没有购买的意愿；对于品牌 3：11、12 号顾客有购买的意愿，13、14、15 号顾客没有购买的意愿**对模型进行优化。

5.4.1 决策树模型的优化

对于决策树模型，根据图 9易得品牌 1 中对于城市居住年龄具有较大的影响，而后是电池、车贷占比、房贷占比。选择品牌 1 中的 3 号顾客，相对来说城市居住年龄较为饱和。相对来说车贷，房贷比较少，具有很大的提高空间，同时，电池、经济的满意度也不高，因此建议建提高对电池，经济的满意度，各提高 2.5%，发现最后结果预测结果变为有购买意愿。

图 11中，品牌 2 中的 8 号顾客，在决策树中购买意愿占比 0.4，可在外观，动力，安全，经济，舒适，电池六个因素中选五个提升，根据决策树中不同因素的重要程度，选择外观、舒适、电池、经济、动力各增加 1%，发现最后结果预测结果变为有购买意愿。

图 13中，品牌 3 选 13 号顾客，电池，品质，外观，操控性，动力性，安全性，经济型，舒适性，八个因素中选五个进行提升，根据决策树中不同因素的重要程度，选择电池，品质，外观，操控性，动力性，安全性，经济型，舒适性各增加 0.625%，发现最后结果预测结果变为有购买意愿，符合题意。

如图 20为模型优化后的训练精度和测试精度的图像：对于神经网络模型：根据卷

图 20 优化的决策树模型精度

积神经网络模型，对其进行补充，思路为将 5% 的满意度分割成 500 份 0.01% 进行各个影响因素之间的加权，得到最终结果最大的值。

5.5 问题五

在问题一中，对电动车的经济性满意度不高是没有购买意愿人群的普遍特点，在问题二可能对购买意愿产生影响的因素中，经济性在三个品牌中均有影响；通过对不同购买意愿人群的经济状况分析可知，经济状况较好的客户更容易有购买意愿；同时，从问题三和问题四的结果可知，不同品牌、不同用户的影响因素不同。

根据以上分析，给出如下三方面销售建议：

(1) 加强购车的经济性。完善售后服务，提供保障措施，减少客户在购车时对电动车售后维修、保养等方面高成本的顾虑，提高客户对新能源汽车经济性能好的认可度。

(2) 将市场定位在较高收入人群。在新能源汽车还没有广泛普及的今天，高收入高学历、有社会责任感、倡导绿色出行的人群更容易对新能源汽车产生购买意愿，以该人群为先导，可加快新能源汽车产业的发展。

(3) 制定精准销售模型。对于不同品牌、不同人群，购买意愿有不同的影响因子，没有固定的销售模式，应该具体事例具体分析，精确分析、精确销售。

附　录

附录 1：问题三的 python 代码

```
# Importing the libraries
#导入包
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
#取数据
dataset =pd.read_csv('C:/Users/Administrator/Desktop/chap4.csv')
#这里取桌面的chap4的代码
#这里取D到Q列
X =dataset.iloc[:,3:16].values
#这里取S列
y =dataset.iloc[:,18].values

# Splitting the dataset into the Training set and Test set
#将数据集拆分为训练集和测试集
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X, y, test_size =0.25,
                                          random_state =0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler
sc_X =StandardScaler()
X_train =sc_X.fit_transform(X_train)
X_test =sc_X.transform(X_test)

# Fitting Classifier to the Training set
#训练集的分类器拟合
from sklearn.tree import DecisionTreeClassifier
classifier =DecisionTreeClassifier(criterion ='entropy', random_state=0,
                                          class_weight={0:1,1:10})
#这里的class_weight={0:1,1:10},max_depth=20参数可以不用加入
# classifier = DecisionTreeClassifier(criterion = 'entropy', random_state=0)
classifier.fit(X_train,y_train)

# Predicting the Test set results
#预测测试集结果
y_pred =classifier.predict(X_test)

# Making the Confusion Matrix
#制作混淆矩阵
```

```
from sklearn.metrics import confusion_matrix
cm =confusion_matrix(y_test, y_pred)

scoretest =classifier.score(X_test,y_test)

from IPython.display import Image
import pydotplus
from sklearn import tree
# 可视化决策树
dot_data =tree.export_graphviz(classifier, out_file=None,
filled=True, rounded=True,
special_characters=True)
graph =pydotplus.graph_from_dot_data(dot_data)
# 显示图片
# graph.get_nodes()[7].set_fillcolor("#FFF2DD")
graph.write_png("C:/Users/Administrator/Desktop/out.png")
```

```
*决策树.
TREE
购买意愿[n]
BY
城市居住龄[s]
/ TREE
DISPLAY =TOPDOWN
NODES =STATISTICS
BRANCHSTATISTICS =YES
NODEDEFS =YES
SCALE =AUTO
/ DEPCATEGORIES
USEVALUES =[.0 1.0]
TARGET =[1.0]
/ PRINT
MODELSUMMARY
CLASSIFICATION
RISK
/ GAIN
CATEGORYTABLE =YES
TYPE =[NODE]
SORT =DESCENDING
CUMULATIVE =NO
/ SAVE
NODEID
PREDVAL
PREDPROB
ASSIGNMENT
/ METHOD
TYPE =CRT
MAXSURROGATES =AUTO
```

```
PRUNE =NONE
/ GROWTHLIMIT
MAXDEPTH =AUTO
MINPARENTSIZE =400
MINCHILDSIZE =200
/ VALIDATION
TYPE =SPLITSAMPLE(75)
OUTPUT =BOTHSAMPLES
/ CRT
IMPURITY =GINI
MINIMPROVEMENT =0.0001
/ COSTS
EQUAL
/ PRIORS
FROMDATA
ADJUST =NO

/ COSTS
EQUAL
COMPUTE
filter_$=(品牌类型=2).
VARIABLE
LABELS
filter_$ '品牌类型=2 (FILTER)'.
VALUE
LABELS
filter_$ 0
'Not Selected'
1
'Selected'.
FORMATS
filter_$ (f1.0).
FILTER
BY
filter_$.
EXECUTE.

*决策树.
TREE
购买意愿[n]
BY
城市居住龄[s]
/ TREE
DISPLAY =TOPDOWN
NODES =STATISTICS
BRANCHSTATISTICS =YES
NODEDEFS =YES
SCALE =AUTO
```

```
/ DEPCATEGORIES
USEVALUES =[.0 1.0]
TARGET =[1.0]
/ PRINT
MODELSUMMARY
CLASSIFICATION
RISK
/ GAIN
CATEGORYTABLE =YES
TYPE =[NODE]
SORT =DESCENDING
CUMULATIVE =NO
/ SAVE
NODEID
PREDVAL
PREDPROB
ASSIGNMENT
/ METHOD
TYPE =CRT
MAXSURROGATES =AUTO
PRUNE =NONE
/ GROWTHLIMIT
MAXDEPTH =AUTO
MINPARENTSIZE =400
MINCHILDSIZE =200
/ VALIDATION
TYPE =SPLITSAMPLE(75)
OUTPUT =BOTHSAMPLES
/ CRT
IMPURITY =GINI
MINIMPROVEMENT =0.0001
/ COSTS
EQUAL
/ PRIORS
FROMDATA
ADJUST =NO

/ COSTS
EQUAL

COMPUTE
filter_$=(品牌类型=3).
VARIABLE
LABELS
filter_$ '品牌类型=3 (FILTER)'.
VALUE
LABELS
filter_$ 0
```

```
'Not Selected'
1
'Selected'.
FORMATS
filter_$ (f1.0).
FILTER
BY
filter_$.
EXECUTE.

*决策树.
TREE
购买意愿[n]
BY
城市居住龄[s]
/ TREE
DISPLAY =TOPDOWN
NODES =STATISTICS
BRANCHSTATISTICS =YES
NODEDEFS =YES
SCALE =AUTO
/ DEPCATEGORIES
USEVALUES =[.0 1.0]
TARGET =[1.0]
/ PRINT
MODELSUMMARY
CLASSIFICATION
RISK
/ GAIN
CATEGORYTABLE =YES
TYPE =[NODE]
SORT =DESCENDING
CUMULATIVE =NO
/ SAVE
NODEID
PREDVAL
PREDPROB
ASSIGNMENT
/ METHOD
TYPE =CRT
MAXSURROGATES =AUTO
PRUNE =NONE
/ GROWTHLIMIT
MAXDEPTH =AUTO
MINPARENTSIZE =400
MINCHILDSIZE =200
/ VALIDATION
TYPE =SPLITSAMPLE(75)
```

```
  OUTPUT =BOTHSAMPLES
  / CRT
  IMPURITY =GINI
  MINIMPROVEMENT =0.0001
  / COSTS
  EQUAL
  / PRIORS
  FROMDATA
  ADJUST =NO

  / COSTS
  EQUAL

\end{appendixx}
\begin{appendixx}
  \section{问题三的神经网络代码}
  from predict import predictToolimport time
  import cv2
  import threadingimport time
  from sensor_md import sensor_obj
  import RPi.GPIO as GPIO
  import time
  from keras.datasets import mnist
  import tensorflow as tf
  from PIL import Image
  import numpy as np
  from tensorflow.keras.utils import to_categorical
  from keras import layers
  from keras import models
  tf.compat.v1.disable_eager_execution()

  model =models.Sequential()
  model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
  model.add(layers.MaxPool2D((2, 2)))
  model.add(layers.Conv2D(64, (3, 3), activation='relu'))
  model.add(layers.MaxPool2D((2, 2)))
  model.add(layers.Conv2D(64, (3, 3), activation='relu'))
  model.add(layers.Flatten())
  model.add(layers.Dense(64, activation='relu'))
  model.add(layers.Dense(10, activation='softmax'))
  # 网络构架

  model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['
                                        accuracy'])
  # 编译步骤

  # 准备图像数据
```

```
train_labels =to_categorical(train_labels)
test_labels =to_categorical(test_labels)
# 准备标签

model.fit(train_images, train_labels, epochs=5, batch_size=64)
test_loss, test_acc =model.evaluate(train_images, train_labels)
print('test_loss:%.6f' % test_loss)
print('test_acc:%.6f' % test_acc)


model.save(r'D:\Python for windows\homewor\cnn_mnist.pkl')

GPIO.setwarnings(False)GPIO.setmode(GPIO.BCM)#
GPIO.setup(4,GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(12,GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(13,GPIO.OUT, initial=GPIO.LOw)
GPIO.setup(16,GPIO.OUT, initial=GPIO.LOw)pin_avoid_obstacle =24
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin_avoid_obstacle, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
def thread(cap):
global ex
ite =0
while True:
global frame
ret, frame =cap.read()
cv2.imshow("capture", frame)
if cv2.waitKey(100) & OxFF ==ord('q'):
ex =True
break
cap.release()
cv2.destroyAllWindows()
pass

if __name__ =='_main_ ':
ex =False
img_pre =predictTool()
cap =cv2.VideoCapture(o)
threa =threading.Thread(target=thread,args=(cap,))
threa.start()
time.sleep(2)
global frame
while True:
status =GPIO.input(pin_avoid_obstacle)
print("zhaugntai", status)
if not status:
time.sleep(0.5)
_, frame =cap.read()
resized_img =cv2.resize(frame, (224, 224), interpolation=cv2.INTER_LINEAR)
```

```
resized_img_rgb =cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB)
id =img_pre.img_predict(resized_img_rgb)
print(id)
if 0 <=id <=5:
GPIO.output(4, GPIO.HIGH)
time.sleep(1)
GPIO.output(4, GPIO.LOw)
elif 6 <=id <=12:
GPIO.output(12, GPIO.HIGH)
time.sleep(1)
GPIO.output(12, GPIO.LOW)
elif 13 <=id <=34 and id !=28:
GPIO.output(13, GPIO.HIGH)
time.sleep(1)
GPIO.output(13, GPIO.LOw)
else:
GPIO.output(16, GPIO.HIGH)
time.sleep(1)
GPIO.output(16, GPIO.LOw)
else:
time.sleep(0.5)
print("wu")
pass
if ex:
break


import os
import json
import torch
from PIL import lmage
from torchvision import transforms
from models.model import resnet101import time
BASE_DIR= os.path.dirname(os.path.abspath( file__))
device =torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
class predictTool():
def _init_(self):
self.path_state_dict =os.path.join(BASE_DIR, "fine_pth" , "resNet101.pth")
self.path_classnames =os.path.join(BASE_DIR, "class _json", "class_indices.json"
                                        )self.num_classes =38
self._data_transform =transforms.Compose(
[transforms.ToTensor(),
transforms.Normalize([0.485,0.456,0.406],[0.229,0.224,0.225])])self.model =self.
                                        __getModel()
self.class_indict =self._load_class_names()
pass
def _getModel(self):
model =resnet101(num_classes=self.num_classes).to(device)
```

```
assert os.path.exists(self.path_state_dict), "file: 't' dose not exist.".format(
                                        self.path_state_dict)
model.load_state_dict(torch.load(self.path_state_dict, map_location=device))
model.to(device)
model.eval()
return model

def img_predict(self, img):
img =self.__img_process(img)
with torch.no_grad():
time_tic =time.time()
output =torch.squeeze(self.model(img.to(device))).cpu()
predict =torch.softmax(output, dim=0)
predict_cla =torch.argmax(predict).numpy()
time_toc =time.time()
print(time_toc -time_tic) # 预测时间
print_res ="class: prob: {:.3}]".format(self.class_indict[str(predict_cla)],
                                        predict[predict_cla].numpy())
print(print_res)
return int(predict_cla)
```

附　录

附录 1：问题四的代码

```
import time
def _init_(self,trig,echo):
self.trig =trig
self.echo =echo
GPIO.setmode(GPIO.BCM)
GPIO.setup(trig,GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(echo, GPIO.IN)pass
def checkdist(self):
try:
GPIO.output(self.trig,GPIO.HIGH)
time.sleep(0.000020)
GPIO.output(self.trig,GPIO.LOW)
while not GPIO.input(self.echo):
pass
t1 =time.time()
while GPIO.input(self.echo):
pass
t2 =time.time()
except:
pass
return ((t2 -t1)* 340/2)*100
if _name__=="_main_":
while(True):
time.sleep(0.5)
k =sensor_obj(22,17)
d =k.checkdist()
print(d)
```