

# 目录

<b>1</b>	<b>问题重述</b>	<b>2</b>
1.1	问题背景 . . . . .	2
1.2	问题提出 . . . . .	2
<b>2</b>	<b>符号定义</b>	<b>3</b>
<b>3</b>	<b>原理概述</b>	<b>3</b>
3.1	线性回归 . . . . .	3
3.2	逻辑回归 . . . . .	3
3.3	决策树 . . . . .	4
3.4	拉格朗日乘子法 . . . . .	4
3.5	GradientBoostingClassifier . . . . .	5
<b>4</b>	<b>模型的建立和求解</b>	<b>5</b>
4.1	问题 1 . . . . .	5
4.1.1	问题分析 . . . . .	5
4.1.2	数据预处理 . . . . .	6
4.1.3	模型建立 . . . . .	6
4.2	问题 2 . . . . .	9
4.2.1	问题分析 . . . . .	9
4.2.2	模型建立 . . . . .	10
<b>5</b>	<b>模型的验证与评价</b>	<b>10</b>
5.1	问题一 . . . . .	10
5.2	问题二 . . . . .	13
<b>6</b>	<b>参考文献</b>	<b>13</b>
<b>7</b>	<b>附录</b>	<b>13</b>

# 1 问题重述

## 1.1 问题背景

近年来，国际保险行业稳步开展，机动车辆保险在我国的财险保费中所占比重最大，以千亿元计。并且，由于我国汽车保有量的继续增加和相关车险的政策出台，投保率也呈继续上升趋势。车险一般可占财险公司业务的 70% 到 80%，所以车险市场历来是财险公司的兵家必争之地。以往，财险公司为了赢得市场，往往采取低价、折扣来争抢客户。但是激烈的市场竞争也带来了利润率的下降，甚至有些企业在亏本经营。大多数车企为了提高利润率开始重视承保车辆的质量。重投保车辆质量的做法，其实是险企科学发展的重要体现，是市场竞争下的企业合理行为。中国目前的车险费率制度，大多数符合“从车主义”。即车险保费多少，主要取决于这辆车本身的各项情况，如车的购置价、座位数、排量、购车年限等，根据这些数据计算出一个基本的车险保费价格，再根据这辆车的上年理赔次数来打不同的折扣。这就导致了中国的车险定价模式非常的单调，相似情况的车型，保费也都差不多。

可以预见未来车险行业的几大发展趋势：

### 1. 车险价格与驾驶行为密切相关

未来的车险定价将逐渐转变为“从人主义”。车险的定价因素将直接与驾驶人的驾驶习惯与行驶里程挂钩，通过驾驶行为来判定车险价格，可能会使车险由原来的一年买一次变成可以一个月买一次。一个具有良好驾驶习惯的车主，可能只需要支付原本保费的 30% 左右，而驾驶习惯不佳的车主，则会在原本保费的基础上继续上涨。

### 2. 同价位车型车险价格完全不同

国内传统的汽车保险定价，通常是以车型和其购置价为主要依据。未来中国车险业，同样的一款车，不同的人开，保费价格会完全不同。这个不同可能是取决于投保人本身的驾驶行为，还可能会以投保人本身的年龄、职业、家庭状况等信息为标准。信息时代的到来，为车险企业提供了一个更加有力的武器，可以通过数字化技术来更加精准地了解客户，制定营销和服务方案。

## 1.2 问题提出

1. 请建立合理的数学模型，对附件一中提供的客户进行精准画像，给出客户的续保概率。
2. 请针对不同的客户设计不同的优惠和福利方案，以提高续保概率。

## 2 符号定义

符号	含义
$x_1$	三者保险额
$x_2$	签单保费
$x_3$	渠道
$x_4$	是否投保车损
$x_5$	是否投保盗抢
$x_6$	是否投保车上人员
$x_7$	投保类别
$x_8$	是否本省车牌
$x_9$	使用性质
$x_{10}$	车辆种类
$x_{11}$	车辆用途
$x_{12}$	险种
$x_{13}$	风险类别
$x_{14}$	客户类别
$x_{15}$	被保险人性别
$x_{16}$	NCD
$x_{17}$	车龄
$x_{18}$	被保险人年龄
$x_{19}$	立案件数
$x_{20}$	已决赔款
predict()	预测模型
$X$	训练样本

## 3 原理概述

### 3.1 线性回归

在统计学中，线性回归 (Linear Regression) 是利用称为线性回归方程的最小平方法函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。这种函数是一个或多个称为回归系数的模型参数的线性组合。只有一个自变量的情况称为简单回归，多个自变量情况的叫做多元回归。我们的建模过程用到的是多元线性回归模型，以下是多元线性回归的公式的假设函数：

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n \quad (1)$$

简化形式：

$$h_{\theta}(x) = \theta^T X \quad (2)$$

其中  $X$ ,  $\theta$  都是一个  $n$  阶向量。

### 3.2 逻辑回归

逻辑回归是一种广义线性模型，是一种用于解决二分类问题的机器学习方法。逻辑回归以线性回归作为理论支持，在线性回归的基础上通过 sigmoid 函数引入非线性元素，因此可以轻松解决 0/1 分类问题。引入 Sigmoid 函数，也称为逻辑函数：

$$g(z) = \frac{1}{1 + e^{-z}} \quad (3)$$

逻辑回归的假设函数形式如下：

$$h_{\theta}(x) = g(\theta^T x) \quad (4)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

由公式 (2)(4) 得到:

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}} \quad (5)$$

其中  $x$  是我们的输入,  $\theta$  为我们要求取的参数,  $x$  和  $\theta$  都是  $n$  维向量。

### 3.3 决策树

决策树是机器学习中的一个预测模型, 代表了对象属性与对象值之间的一种映射关系。决策树是一种树形结构, 其中每个内部节点表示一个属性上的测试, 每个分支代表一个测试输出, 每个叶节点代表一种类别。决策树学习基本算法如下:

```

输入: 训练集  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ ;
      属性集  $A = \{a_1, a_2, \dots, a_d\}$ .
过程: 函数 TreeGenerate( $D, A$ )
1: 生成结点 node;
2: if  $D$  中样本全属于同一类别  $C$  then
3:   将 node 标记为  $C$  类叶结点; return
4: end if
5: if  $A = \emptyset$  OR  $D$  中样本在  $A$  上取值相同 then
6:   将 node 标记为叶结点, 其类别标记为  $D$  中样本数最多的类; return
7: end if
8: 从  $A$  中选择最优划分属性  $a_*$ ;
9: for  $a_*$  的每一个值  $a_v^*$  do
10:  为 node 生成一个分支; 令  $D_v$  表示  $D$  中在  $a_*$  上取值为  $a_v^*$  的样本子集;
11:  如果  $D_v$  为空 then
12:    将分支结点标记为叶结点, 其类别标记为  $D$  中样本数最多的类; return
13:  else
14:    以 TreeGenerate( $D_v, A \setminus \{a_*\}$ ) 为分支结点
15:  end if
16: end for
输出: 以 node 为根结点的一棵决策树
    
```

Figure 1: 决策树基本学习算法

显然, 决策树的生成是一个递归过程。在决策树基本算法中, 有三种情况会导致递归返回: (1) 当前节点包含的样本全属于同一类别, 无需划分; (2) 当前属性集为空, 或是所有样本在所有属性上取值相同无法划分; (3) 当前节点包含的样本集合为空, 不能划分。由 Figure 1 的算法可以看出, 决策树学习的关键是如何选择最优划分属性。一般而言, 随着划分过程不断进行, 我们希望决策树的分支节点所包含的样本尽可能属于同一类别, 即结点的“纯度”越来越高。“信息熵”是度量样本集合纯度最常用的一种指标。假定当前样本集合  $D$  中第  $k$  类样本所占的比例为  $p_k$ , ( $k = 1, 2, \dots, |y|$ ), 则  $D$  的信息熵定义为:

$$Ent(D) = - \sum_{k=1}^{|y|} p_k \log_2 p_k \quad (6)$$

$Ent(D)$  的值越小, 则  $D$  的纯度越高。

### 3.4 拉格朗日乘子法

拉格朗日乘子法 (Lagrange multipliers) 是一种寻找多元函数在一组约束下的极值的方法。通过引入拉格朗日乘子, 可以将  $d$  个变量与  $k$  个约束条件的最优化问题转化为具有  $d+k$  个变量的无约束优化问题求解。

引入拉格朗日乘子  $\vec{\lambda} = (\lambda_1, \dots, \lambda_m)^T$  和  $\mu = (\mu_1, \dots, \mu_n)^T$ , 相应的拉格朗日函数为:

$$L(\vec{x}, \vec{\lambda}, \vec{\mu}) = f(\vec{x}) + \sum_{i=1}^m \lambda_i h_i(\vec{x}) + \sum_{j=1}^n \mu_j g_j(\vec{x})$$

有不等式约束条件引入的 KKT 条件 ( $j=1, 2, \dots, n$ ) 为:

$$\begin{cases} g_j(\vec{x}) = 0; \\ \mu_j = 0; \\ \mu_j g_j(\vec{x}) = 0 \end{cases} \quad (7)$$

拉格朗日”对偶函数“(dual function) $\Gamma: R \times R \mapsto R$  定义为:

$$\Gamma(\vec{\lambda}, \vec{\mu}) = \inf_{f \in D} L(\vec{x}, \vec{\lambda}, \vec{\mu}) = \inf_{f \in D} (f(\vec{x}) + \sum_{i=1}^m \lambda_i h_i(\vec{x}) + \sum_{j=1}^n \mu_j g_j(\vec{x})) \quad (8)$$

### 3.5 GradientBoostingClassifier

梯度上升决策树模型是 boost 提升算法和决策树模型的结合, Gradient Boosting 采用和 AdaBoost 同样的加法模型, 在第  $m$  次迭代中, 前  $m-1$  个基学习器都是固定的, 即:

$$f_m(x) = f_{m-1}(x) + \rho_m h_m(x)$$

因而在第  $m$  步我们的目标是 minimize 损失函数  $L(f) = \sum_{i=1}^N (y_i, f_m(x_i))$ , 进而求得相应的基学习器。若将  $f(x)$  当成参数, 则同样可以使用梯度下降法:

$$f_m(x) = f_{m-1}(x) - \rho_m \cdot \frac{\partial}{\partial f_{m-1}(x)} L(y, f_{m-1}(x))$$

对比上面两个式子, 可以发现若将  $h_m(x) \simeq -\frac{\partial L(y, f_{m-1}(x))}{\partial f_{m-1}(x)}$ , 即用基学习器  $h_m(x)$  拟合前一轮模型损失函数的负梯度, 就是通过梯度下降法最小化  $L(f)$ 。由于  $f(x)$  实际为函数, 所以该方法被认为是函数空间的梯度下降。负梯度也被称为“响应 (response)”或“伪残差 (pseudo residual)”, 从名字可以看出是一个与残差接近的概念。直觉上来看, 残差  $r = y - f(x)$  越大, 表明前一轮学习器  $f(x)$  的结果与真实值  $y$  相差较大, 那么下一轮学习器通过拟合残差或负梯度, 就能纠正之前的学习器犯错较大的地方。

## 4 模型的建立和求解

### 4.1 问题 1

#### 4.1.1 问题分析

用户画像是一种全面勾画用户、联系用户与产品的良好工具, 是建立在一系列真实数据之上的用户目标模型。用户画像描绘用于标识用户的各项信息, 比如用户背景、身份、兴趣、需求、心理、性格、收入、职业等, 通过用户画像, 我们可以细致地了解到一个用户的信息全貌。随着互联网技术的发展和大数据时代的来临, 在数据驱动下, 用户画像的内涵和外延都发生了变化, 主要是通过数据刻画用户特征, 从而为用户提供优质服务。第一小题要求根据附件一的数据对用户进行精确画像, 建立合理的信息模型, 给出客户的续保概率, 认真理解题意, 不难发现我们要做的其实是从给定的数据集中提取出用户的个人信息集合, 来独立进行用户需求、偏好和兴趣描述的模式。

提取用户标签是刻画用户画像的关键。刻画用户画像要求我们从繁杂的数据中抽取共同的特征值, 根据特征值对群体进行定义, 一个群体会有多个标签, 不同的群体之间也会有标签的重合, 此时标签的权重反映了不同群体的核心特征。应用大数据分析、数据挖掘、机器学习等技术处理和分析用户的行为数据和信息数据是提取用户画像的偏好、兴趣标签的重要途径之一。对于附件一中的数据, 我们采用 k-mean 聚类算法来提取用户标签, 在此之前, 首先对附件一中的数据做如下预处理:

1. “分类”数据离散化; 如将附件一中“渠道”的值分别对应数值 0-7;
2. 对连续型数据进行“归一化”处理, 提高模型训练时的收敛速度, 提高模型性能;

在完成数据的预处理过程之后, 我们将附件一中不同的属性分别用 k-mean 算法进行聚类, 观察聚类效果, 将聚类效果明显的属性提取为我们的用户标签。在确定用户标签之后, 我们通过用户标签将附件一中的数据划分为不同类别的数据, 针对每一个类别分别使用不同的模型对数据进行建模, 选取预测效果最好的模型作为该类别的预测模型。

销售渠道	数值	保单性质	数值	投保类别	数值	是否本省车牌	数值
车商渠道	0	续保	0	单交强	0	否	0
电网销	1	转保	1	单商业	1	临时	1
个人代理	2			交商全保	2	是	2
交叉销售	3						
门店	4						
普通兼代	5						
直拓	6						
专业中介	7						

#### 4.1.2 数据预处理

首先我们将附件一中的非数字的属性值离散化，对应关系如下（此处只列举部分）：

使用性质	数值	车辆种类	数值	车辆用途	数值
城市公交	0	10 吨及 10 吨以上挂车	0	10 座以上客车	0
出租租赁	1	10 吨及 10 吨以下货车	1	9 座及 9 座以下非营运客车 (含越野)	1
党政机关、事业团体用车	2	10 座及 20 座以下客车	2	9 座以上非营运客车	2
非营运货车	3	20 座及 36 座以下客车	3	9 座以下客车	3
公路客运	4	2 吨及 5 吨以下货车	4	出租车	4
家庭自用车	5	2 吨以下货车	5	带拖挂的载货汽车	5
企业非营业用车	6	36 座及 36 座以上客车	6	带拖挂汽车	6
特种车	7	5 吨及 10 吨以下挂车	7	低速货车和三轮汽车	7
营业货车	8	5 吨及 10 吨以下货车	8	矿山专用车	8
		6 座及 10 座以下客车	9	矿山作业用车	9
		6 座以下客车	10	旅游载客汽车	10
		低速载货汽车	11	其他车辆	11
		特种车二	12	轻微型载货汽车	12
		特种车二挂车	13	微型载货汽车	13
		特种车三	14	重、中型载货汽车	14
		特种车四	15	空白	15

Figure 2: 属性离散化

注意，上述对类别特征编码的过程中的数值  $n$  并不是指代该种类别的取值设定为  $n$ ，这里我们采用的是 one-hot 编码规则。举例来说，对属性车辆用途来说， $n = 7$  代表的是一个 16 行 1 列的列向量，其中  $x_6 = 0$ ，且  $x_i \neq 0$  when  $i \neq 6$ 。我们采用 one-hot 编码规则的好处是对于不能直接在属性值上计算距离的“无序属性”来说，使用 one-hot 编码规则能够使得使用 k-mean 算法得到的任意两个不同的类之间的距离都为 1，从而消除距离对聚类结果的影响。有一点需要注意的是，对于某些属性来说，不同类别之间的差异不能全部简单地设置为 1，因为这些类别之间的差异代表着不同的现实意义，比如，对于属性 NCD 的三个可能取值连续 3+ 无事故连续两年无事故上年未发生有责事故来说，“连续 3+ 无事故”和“连续两年无事故”所代表的驾驶能力的差别明显不同于“连续两年无事故”和“上年未发生有责事故”的驾驶能力差异。因此，对于此类的属性，我们不能简单地将其视为离散变量，而应该区分其等级。

#### 4.1.3 模型建立

首先将数据中的不同属性分别用 k-mean 算法进行聚类，将聚类效果明显的属性提取为我们的用户标签。当然，用户标签的提取还需要结合一定的先验知识。由于附件中属性的个数太多，这里我们只列举出聚类效果较好的几个属性，如下图所示：可以看到，以上几个属性的聚类效果都比较好。将以上几个属性提取为用户标签，提取的用户标签主要包括以下四个内容：

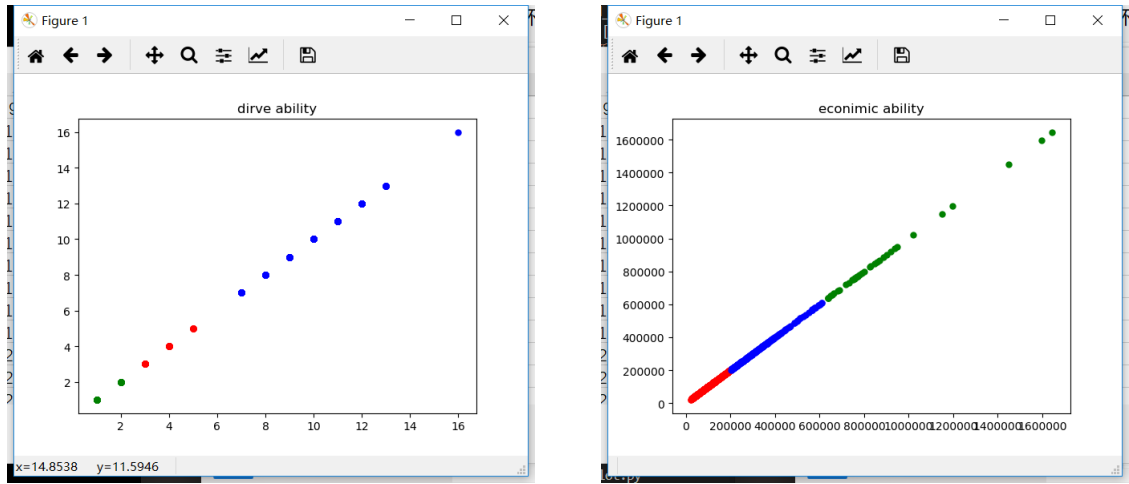


Figure 3: 经济水平聚类、驾驶习惯 (技术) 聚类

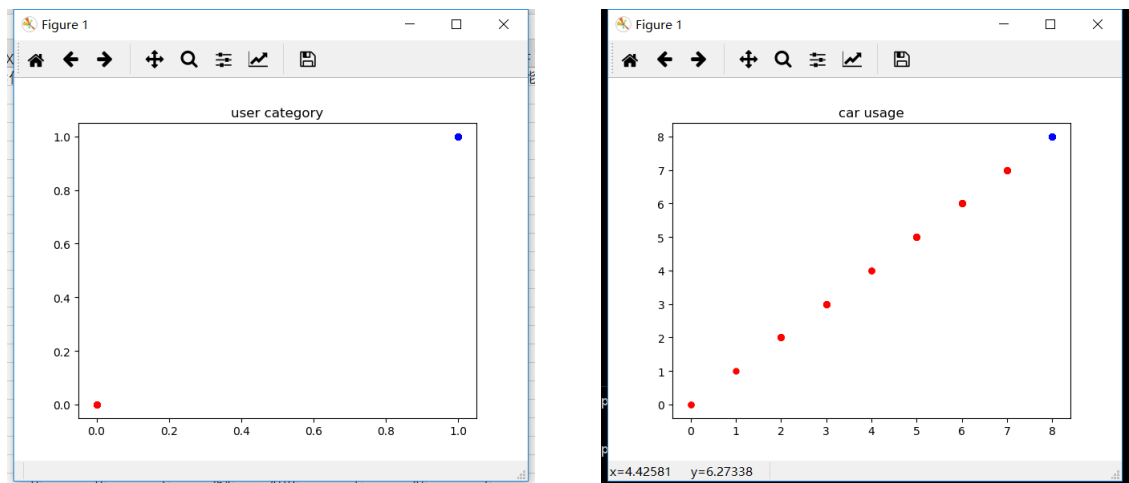


Figure 4: 用户类别聚类与车辆用途聚类

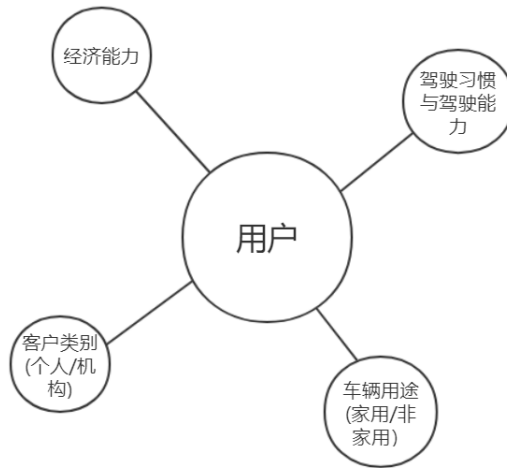


Figure 5: 用户画像

标签的取值集合为:

标签名称	分类
用户类别	机构
	个人
车辆用途	家庭用车
	非家庭用车
经济能力	富有(新车购置价 $\geq 147800$ )
	小康(81300~147800)
	一般(0~81300)
驾驶技术与驾驶习惯 (根据NCD与)	良好
	中等
	差

Figure 6: 标签的取值集合

确定好用户标签之后，我们将附件一的数据按照不同的类别进行分类。由于对于不同的分类类别可能适合使用不同的模型进行训练，所以将分类之后的数据分别使用逻辑回归分类器，决策树分类模型，GradientBoostingClassifier 进行训练，挑选预测结果最好的模型作为该类别的学习模型。我们以模型的”准确率 (precision) “和”召回率 (recall) “作为预测的指标，由于根据用户标签进行分类得到的类别有很多，这里我们只给出类别 A “个人用户、驾驶习惯中等、家庭用车、经济能力小康” 的结果，如下所示：

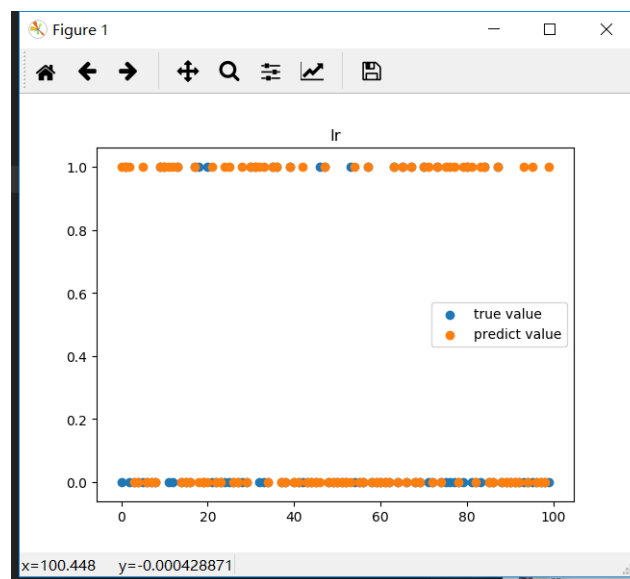


Figure 7: logistic 回归模型的预测结果



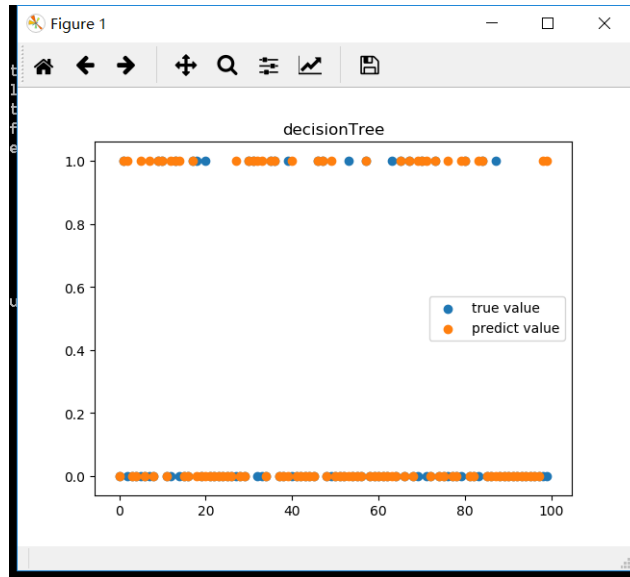


Figure 8: 1 决策树模型的预测结果

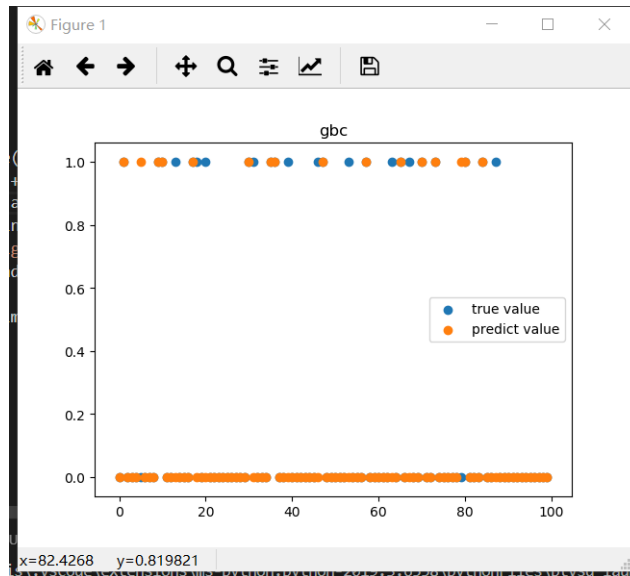


Figure 9: GradientBoostingClassifier 的预测结果

可以看到，对于类别 A 来说，logistic 回归模型和决策树模型的预测效果要更好一些，具体使用哪一个模型应根据具体情况决定。

## 4.2 问题 2

### 4.2.1 问题分析

针对不同的用户设计不同的优惠和福利方案要求我们对问题 1 中划分出来的所有用户类别建立各自的数学模型，模型的目标函数是最大化该类用户的续保概率。自变量为“不同的优惠和福利方案”，即保险公司可以调控的与用户相关的属性，对附件一中数据的属性进行筛选，不难发现可以用来调节以提高续保概率的因素只有“渠道”、“签单保费”、“险种”、“是否投保车损”、“是否投保车上人员”和“三者险保额”。所以“为不同的用户设计不同的优惠和福利方案”实际上要求我们调节以上属性值来尽量增大用户续保的概率。不难发现这其实是一个最优化问题，这里我们引入拉格朗日乘子法来求解这个它。

#### 4.2.2 模型建立

我们的目标函数为：

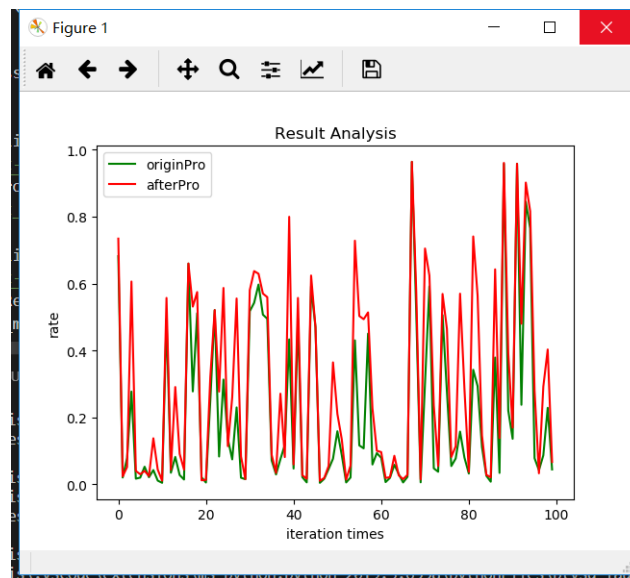
$$\min(1 - \text{predict}(X))$$

约束条件为：

$$s.t. \begin{cases} 0 \leq x_1 \leq 1000000 \\ 0 \leq x_2 \\ 0 \leq x_3 \leq 7 \\ 0 \leq x_4 \leq 1 \\ 0 \leq x_5 \leq 1 \\ 0 \leq x_6 \leq 1 \end{cases}$$

注意，这里假设  $x_3, x_4, x_5, x_6$  均是连续变量。欲提高每个用户类别的续保概率，我们只要求得每个类别对应的数学模型中目标函数的最优解即可。求解最优解可使用拉格朗日乘子法，求解最优解时的  $x_1$  到  $x_6$  的取值即确定了我们的优惠方案 (假若  $x_1$  10000 20000 “ ” 10000 )  
到  $x_6$  中存在非整数值的应该对其取整。(属性的值不能是连续的，之前假设自变量为连续的是为便于对模型进行训练)。由于类别太多，我们只选取一个类别使用我们的模型进行分析，结果如下：

$x_1$



上图中，绿色的曲线代表的是初始数据预测到的续保概率，红色曲线代表通过调节属性之后预测到的用户续保概率，从图中可以直关地看出，通过调节一些属性，我们可以做到提高用户的续保概率 (推行优惠方案和福利)。

## 5 模型的验证与评价

### 5.1 问题一

验证我们提取的用户标签是否能够提高对客户续保概率的预测准确性，下图是直接对附件一中的数据  
进行建模的预测结果：

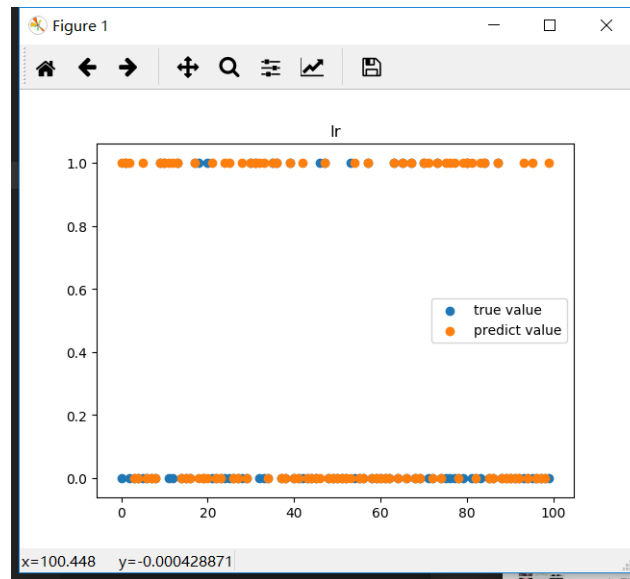


Figure 10: logistic 回归模型的预测结果

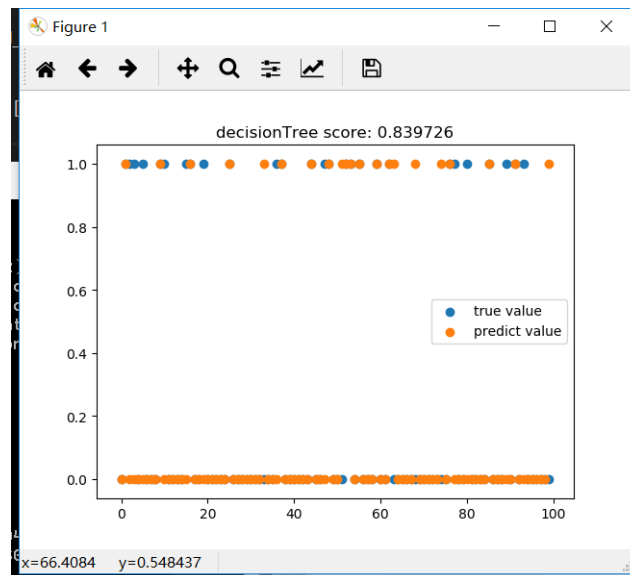


Figure 11: 决策树模型的预测结果

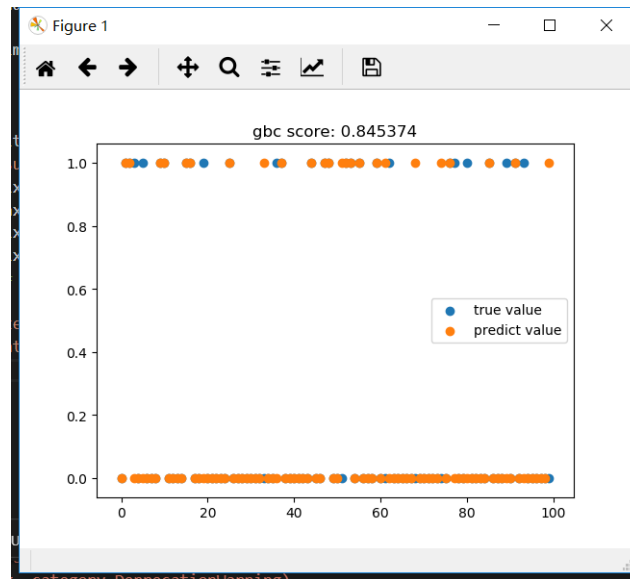


Figure 12: GradientBoostingClassifier 模型的预测结果

我们将使用用户标签进行分类后再进行预测的结果与直接使用附件一中的所有属性进行预测的结果进行对比如下：

使用用户标签：

lr			
	precision	recall	support
不续保	0.84	0.94	10498
续保	0.55	0.29	2603
aver/total	0.78	0.81	13101

decision tree			
	precision	recall	support
不续保	0.86	0.95	10498
续保	0.67	0.39	2603
aver/total	0.82	0.84	13101

gbc			
	precision	recall	support
不续保	0.87	0.95	10498
续保	0.67	0.42	2603
aver/total	0.83	0.84	13101

初始数据：


lr			
	precision	recall	support
不续保	0.92	0.68	1511
续保	0.41	0.78	428
aver/total	0.8	0.7	1939

decisiontree			
	precision	recall	support
不续保	0.91	0.78	1511
续保	0.48	0.72	428
aver/total	0.81	0.77	1939

gbc			
	precision	recall	support
不续保	0.86	0.95	1511
续保	0.73	0.43	428
aver/total	0.83	0.84	0.82

## 5.2 问题二

由于是预测模型，无法验证其结果的正确性，这里我们给出每种类别的优惠方案如下：

1	三者险保额	签单保费	渠道	是否投保车损	是否投保盗抢	是否投保车上人员
原方案	0	806.6	电商渠道	不投保车损	不投保盗抢	不投保车上人员
推荐方案	0	806.6	电商渠道	不投保车损	不投保盗抢	不投保车上人员
2	三者险保额	签单保费	渠道	是否投保车损	是否投保盗抢	是否投保车上人员
原方案	500000	3513.1	交叉销售	投保车损	不投保盗抢	不投保车上人员
推荐方案	500000	3513.1	电网销	投保车损	不投保盗抢	不投保车上人员
3	三者险保额	签单保费	渠道	是否投保车损	是否投保盗抢	是否投保车上人员
原方案	1000000	1318	电网销	不投保车损	不投保盗抢	不投保车上人员
推荐方案	1000000	1318	电网销	不投保车损	不投保盗抢	不投保车上人员
4	三者险保额	签单保费	渠道	是否投保车损	是否投保盗抢	是否投保车上人员
原方案	500000	4175.7	电网销	投保车损	不投保盗抢	不投保车上人员
推荐方案	500000	4175.7	电网销	投保车损	不投保盗抢	不投保车上人员
5	三者险保额	签单保费	渠道	是否投保车损	是否投保盗抢	是否投保车上人员
原方案	300000	1124	普通兼代	不投保车损	不投保盗抢	不投保车上人员
推荐方案	300000	1124	电网销	不投保车损	不投保盗抢	不投保车上人员
6	三者险保额	签单保费	渠道	是否投保车损	是否投保盗抢	是否投保车上人员
原方案	1000000	7783	专业中介	投保车损	不投保盗抢	不投保车上人员
推荐方案	1000000	7783	电网销	 (Ctrl) ▾	不投保盗抢	不投保车上人员
7	三者险保额	签单保费	渠道	是否投保车损	是否投保盗抢	是否投保车上人员
原方案	500000	1190	交叉销售	不投保车损	不投保盗抢	不投保车上人员
推荐方案	500000	1190	电网销	不投保车损	不投保盗抢	不投保车上人员

## 6 参考文献

### References

- [1] 作者：百度百科词条. 多元线性回归.  
<https://baike.baidu.com/item/%E5%A4%9A%E5%85%83%E7%BA%BF%E6%80%A7%E5%9B%9E%E5%BD%92/10702248?fr=aladdin>.
- [2] 周志华. 机器学习. 北京: 清华大学出版社, P403-P405 2016.

[3] 作者: 灵夕 | 月下灵 <https://blog.csdn.net/xiaolong42/article/details/80879337> [https](https://blog.csdn.net/xiaolong42/article/details/80879337) :

[4] 作者: 人民网 <http://sh.people.com.cn/n2/2017/0910/c134768-30715425.html>

## 7 附录

**fir.py** from sklearn import preprocessing; import numpy as np; import pandas as pd; import matplotlib.pyplot as plt

```
data = pd.read_excel("zgl.xlsx")
分类变量 classifyVar = ['渠道', '投保类别', '01 本省车牌', '使用性质', '车辆种类', '车辆用途', '险种', '风险类别 (A 最低, E 最高)', '客户类别', '被保险人性别', '是否投保车损', '是否投保盗抢', '是否投保车上人员'] 其余变量 otherVar = ['NCD', '车龄', '被保险人年龄', '三者险保额', '新车购置价', '保单保费', '立案件数', '已决赔款']
```

```
X = data.iloc[:, 0:37] Y = data.iloc[:, 37] print(Y) data = data.drop(columns=[])
```

```
Xclassify = X[classifyVar]
```

```
Xother = X[otherVar]
```

```
进行数据预处理, 对分类变量和普通变量分别进行 One-hot 编码以及归一化 from sklearn import preprocessing ss_x = preprocessing.StandardScaler() oneHenc = preprocessing.OneHotEncoder() Xother = ss_x.fit_transform(Xother) Xclassify = oneHenc.fit_transform(Xclassify)
```

```
from scipy.sparse import hstack X = hstack((Xother, Xclassify))
```

```
划分训练集和验证集 from sklearn.cross_validation import train_test_split, cross_val_score X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 4)
```

```
from sklearn.metrics import classification_report def try_different_method(model, name): model.fit(X_train, Y_train) score = cross_val_score(model, X_train, Y_train, cv = 5).ravel() score = np.mean(scores) print(model.predict_proba(X_test)) score = model.score(X_test, Y_test) result = model.predict(X_test)
```

```
plt.figure() plt.scatter(np.arange(100), Y_test[:100], label = 'truevalue') plt.scatter(np.arange(100), result[:100], label = 'predictvalue') plt.title(name + ' score :') plt.legend() plt.show()
```

```
print(classification_report(Y_test, result, target_names = [' ', '']))
```

```
from sklearn.linear_model import Perceptron http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron() try_different_method(ppn, 'perceptron')
```

```
from sklearn.linear_model import LogisticRegression http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression(C = 1000.0, random_state = 0) try_different_method(lr, 'lr')
```

```
from sklearn.svm import SVC http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html sklearn.svm.SVC svm = SVC(kernel='linear', C=1.0, random_state = 0, probability = True) try_different_method(svm, 'svc')
```

```
from sklearn.tree import DecisionTreeClassifier http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier criterion='entropy', max_depth = 10, random_state = 0) try_different_method(tree, 'decisionTree')
```

```
from sklearn.ensemble import RandomForestClassifier forest = RandomForestClassifier(criterion='entropy', n_estimators = 10, random_state = 1, n_jobs = 2) http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier
```

```
from sklearn.tree import ExtraTreeClassifier ExtraTree = ExtraTreeClassifier() try_different_method(ExtraTree, 'ExtraTree')
```

```
from sklearn.ensemble import BaggingClassifier Bagging = BaggingClassifier() try_different_method(Bagging, 'Bagging')
```

```
from sklearn.ensemble import GradientBoostingClassifier gbc = GradientBoostingClassifier() try_different_method(gbc, 'gbc')
```

**sec.py** from sklearn import preprocessing; import numpy as np; import pandas as pd; import matplotlib.pyplot as plt

```
data = pd.read_excel("C :
```

```
Users
```

```
artemis
```

```
Desktop
```

```
test
```

```
data2
```

```
3.xlsx")
```

```
分类变量 classifyVar = ['投保类别', '01 本省车牌', '使用性质', '车辆种类', '车辆用途', '险种', '风险类别 (A 最低, E 最高)', '客户类别', '被保险人性别', '渠道', '是否投保车损', '是否投保盗抢', '是否投保车上人员'] 其余变量 otherVar = ['NCD', '车龄', '被保险人年龄', '新车购置价', '立案件数', '已决赔款', '三者险保额', '保单保费']
```

```
X = data.iloc[:, 0:] Y = data.iloc[:, 36] print(Y) data = data.drop(columns=[])
```

```

Xclassify = X[classifyVar]
Xother = X[otherVar]
进行数据预处理, 对分类变量和普通变量分别进行 One-hot 编码以及归一化 from sklearn import pre-
processing ssx = preprocessing.StandardScaler()oneHenc = preprocessing.OneHotEncoder()Xother =
ssx.fittransform(Xother)Xclassify = oneHenc.fittransform(Xclassify)
from scipy.sparse import hstack X = hstack((Xother, Xclassify))
划分训练集和验证集 from sklearn.crossvalidationimporttraintest_split, crossval_scoreXtrain, Xtest, Ytrain, Ytest =
traintest_split(X, Y, testsize = 0.2, randomstate = 4)
from sklearn.metrics import classificationreportdef trydifferent_method(model, name): model.fit(Xtrain, Ytrain)score =
crossval_score(model, Xtrain, Ytrain, cv = 5).ravel()score = np.mean(scores)print(model.predictproba(Xtest))'''score =
model.score(Xtest, Ytest)'''result = model.predict(Xtest)
plt.figure() plt.scatter(np.arange(100), Ytest[: 100], label = ' truevalue')plt.scatter(np.arange(100), result[:
100], label = ' predictvalue')plt.title(name)plt.legend()plt.show()
print(classificationreport(Ytest, result, targetnames = [ ' ', ' ']))
from sklearn.linearmodelimportPerceptronhttp://scikit-learn.org/stable/modules/generated/sklearn.linearmodel.Perceptron()trydifferent_method(ppn, 'perceptron')
from sklearn.linearmodelimportLogisticRegressionhttp://scikit-learn.org/stable/modules/generated/sklearn.linearmodel.LogisticRegression(C = 1000.0, randomstate = 0, classweight = ' balanced')trydifferent_method(lr, 'lr',)
from sklearn.svm import SVC http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.htmlsklearn.svm.SVC
svm = SVC(kernel='linear', C=1.0, randomstate = 0, probability = True)trydifferent_method(svm, 'svc')
from sklearn.tree import DecisionTreeClassifier http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier()
tree = DecisionTreeClassifier(criterion='entropy', maxdepth = 10, randomstate = 0, classweight = '
balanced')trydifferent_method(tree, 'decisionTree')
from sklearn.ensemble import RandomForestClassifier forest = RandomForestClassifier(criterion='entropy',
nestimators = 10, randomstate = 1, njobs = 2, classweight = ' balanced')http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier()
from sklearn.tree import ExtraTreeClassifier ExtraTree = ExtraTreeClassifier(classweight = ' balanced')trydifferent_method(ExtraTree, 'ExtraTree')
from sklearn.ensemble import BaggingClassifier Bagging = BaggingClassifier() trydifferent_method(Bagging, 'Bagging')
from sklearn.ensemble import GradientBoostingClassifier gbc = GradientBoostingClassifier() trydifferent_method(gbc, 'GradientBoosting')
''' 第二问'''
目标样本 index = 0
secondData =pd.readexcel("C:
Users
artemis
Desktop

test
data2
3.xlsx")
secondX = secondData.iloc[:, 0:]
secClassifyVar = [ '投保类别', '01 本省车牌', '使用性质', '车辆种类', '车辆用途', '险种', '风险类
别 (A 最低, E 最高)', '客户类别', '被保险人性别'] secOtherVar = ['NCD', '车龄', '被保险人年龄', '
新车购置价', '立案件数', '已决赔款']
targetVar = [ '三者险保额', '签单保费', '渠道', '是否投保车损', '是否投保盗抢', '是否投保车上
人员']
origin = secondX[targetVar].values
进行数据预处理, 对分类变量和普通变量分别进行 One-hot 编码以及归一化 from sklearn import pre-
processing ssx = preprocessing.StandardScaler()oneHenc = preprocessing.OneHotEncoder()Xother =
ssx.fittransform(Xother)Xclassify = oneHenc.fittransform(Xclassify)
预测模型, 可以进行调整 model = gbc
delta = 0.0001
from scipy.optimize import minimize 优化的目标函数 def func(args): """ 定义目标函数 F(x) """
args = np.array(args) Xclassify = secondX[secClassifyVar].values[index].tolist() Xclassify.append(args[2])
Xclassify.append(args[3]) Xclassify.append(args[4]) Xclassify.append(args[5])
Xother = secondX[secOtherVar].values[index].tolist() Xother.append(args[0]) Xother.append(args[1])
Xother = np.array(Xother) Xclassify = np.array(Xclassify)
Xother = Xother.reshape(-1,1).transpose() Xclassify = Xclassify.reshape(-1,1).transpose()
Xother = ssx.transform(Xother)Xclassify = oneHenc.transform(Xclassify)
from scipy.sparse import hstack testX = hstack((Xother, Xclassify))

```

```

print(testX)
prob = model.predict_proba(testX)print(prob)result = prob[0][0]returnresult * 10000
cons = ('type': 'ineq', 不等式约束条件'fun' : lambda x: x[0] - delta, 'type': 'ineq', 不等式约束条
件'fun' : lambda x: 2000000 - x[0] - delta, 'type': 'ineq', 不等式约束条件'fun' : lambda x: x[1] - delta,
'type': 'ineq', 不等式约束条件'fun' : lambda x: 1 - x[2] - delta, 'type': 'ineq', 不等式约束条件'fun' :
lambda x: x[2] - delta, 'type': 'ineq', 不等式约束条件'fun' : lambda x: 1 - x[3] - delta, 'type': 'ineq', 不
等式约束条件'fun' : lambda x: x[3] - delta, 'type': 'ineq', 不等式约束条件'fun' : lambda x: 1 - x[4] -
delta, 'type': 'ineq', 不等式约束条件'fun' : lambda x: x[4] - delta, 'type': 'ineq', 不等式约束条件'fun' :
lambda x: 1 - x[5] - delta, 'type': 'ineq', 不等式约束条件'fun' : lambda x: x[5] - delta, )
初值
originPro = [] afterPro = [] for i in range(100): index = i + 5000 originPro.append(model.predict_proba(X.toarray())[ind
origin[index]print('originalChioce :')print(Y[index])print(x0)print('originalplan')print(x0)res = minimize(func,x0,
(), constraints = cons, method = 'SLSQP', options = 'disp' : True)afterPro.append(1-res.fun/10000)print('newplan
开始画图 plt.title('Result Analysis') plt.plot(range(100), originPro, color='green', label='originPro')
plt.plot(range(100), afterPro, color='red', label='afterPro') plt.legend() 显示图例
plt.xlabel('iteration times') plt.ylabel('rate') plt.show()

```

```

user.py import numpy as np; import pandas as pd; import matplotlib.pyplot as plt
targetFactor = '新老用户' data = pd.read_excel("real2(2).xlsx")originalVar = [' ' ]otherVar =
[' ' ]
X = data.iloc[:, 2:26] Xoriginal = X[originalVar] Xother = X[otherVar]
进行数据预处理, 对分类变量和普通变量分别进行 One-hot 编码以及归一化 from sklearn import
preprocessing ss_x = preprocessing.StandardScaler()Xother = ss_x.fit_transform(Xother)oneHenc =
preprocessing.OneHotEncoder()Xoriginal = oneHenc.fit_transform(Xoriginal)
from scipy.sparse import hstack X_train = hstack((Xoriginal, Xother))
from sklearn.cluster import KMeans 分类个数 n_clusters = 2kmean = KMeans(n_clusters = n_clusters)kmean.fit(X_train)
originalVar.extend(otherVar) data_list = X[originalVar].valuesoriginalVar.append(targetFactor)
labels = kmean.labels_[:, np.newaxis]data_list = np.hstack((data_list, labels))
strat_df1 = pd.DataFrame(data = data_list, columns = originalVar)
writer = pd.ExcelWriter((targetFactor + '.xlsx')) strat_df1.to_excel(writer, 'Sheet1')writer.save()

```