目录

6 数据离散化

离散化(discretization 或quantization) 也称为分箱(binning), 是一类适用于定量变量的数据变换方法. 同时, 离散化可以减少数据的取值, 因而也是一种数据归约方法. 例如, 作直方图需要先进行数据离散化, 将变量的取值划分为若干个区间. 决策树模型实际上也使用了离散化方法, 选取一定的临界值划分区间.

6.1 离散化的作用与分类

理解离散化的作用, 可以从两个角度展开.

第一个角度, 离散化有助于更好地利用数据信息.

(1) 离散化可以过滤噪声和离群值. 例如, 决策树CART模型不易受自变量的离群值影响, 这正是得益于CART模型对定量变量的离散化处理.

(2) 离散化有助于缺失值信息的利用和处理. 离散化后的变量以区间形式取值, 成为定性变量, 此时可以将缺失值作为一个类别, 从而使用“缺失”这一信息.

(3) 离散化后的变量可保持在相似的衡量尺度上, 使得变量之间具有更好的可比性.

第二个角度, 离散化有助于提升效率、增进理解.

(1) 放入离散化后的自变量, 可以在线性模型框架下获得非线性的效果, 有助于提升模型构建的效率. 例如, 多元自适应样条回归(Multivariate Adaptive Regression Splines, MARS) 即采用了这种思路.

(2) 离散化减少了数值, 离散化后的变量存储与计算效率更高, 使得建模运算更为简单、速度更快. 例如, LightGBM具有较快的运算速度, 与它使用了离散化方法(直方图) 有很大关系.

(3) 离散化具有一定的变量筛选功能. 在有监督离散化中, 自变量经离散化后可能仅包含一个区间, 这表明该自变量对因变量的预测贡献度低, 可以作为变量筛选的依据.

(4) 通过离散化, 以更加粗略的尺度刻画变量信息. 这有助于让模型更加稳定, 并且减小模型过拟合的风险.

(5) 离散化可以增进对模型的理解和解释. 例如, 评分卡模型使用了离散化的变量信息, 它将一定的取值区间对应于一个评分值, 便于业务人员理解和操作.

但是, 数据离散化也存在一定的缺陷. 首先, 离散化处理会损失数据信息. 定量变量经离散化后, 成为定序变量, 但由于我们对定序变量往往没有较好的处理办法, 最终只能将它们看作定类变量. 这样, 量化信息最终变为类别信息, 可能造成较大的信息损失. 其次, 有研究表明, 定性的冗余变量更容易被认定为有用变量. 因此, 定量变量经离散化后成为定性变量, 这种转换可能会影响我们对变量重要性的判断.

离散化方法相当丰富, 其分类方式也有多种.

(1) 按照是否使用了因变量信息, 可划分为有监督(supervised) 和无监督(unsupervised) 离散化. 有监督离散化需要使用后续数据分析中模

型因变量Y的信息(Y必须是定性变量), 无监督离散化则不需要Y的信息.

(2) 按照离散化与模型的关联, 可划分为静态(static) 和动态(dynamic) 离散化. 静态离散化与后续数据分析中的模型无关, 在建模之前完成离散化处理. 动态方法内嵌于模型(如决策树CART模型) 之中, 在模型构建时产生.

(3) 按照区间的产生过程, 可划分为直接(direct) 和递增(incremental) 离散化. 直接离散化也可称为非分层(non-hierarchical) 离散化, 一般先指定划分的区间数量, 然后产生相应数量的区间. 递增离散化也可称分层(hierarchical) 离散化, 往往不能事先确定区间数量, 需要逐步产生区间, 直到满足一定的停止条件.

(4) 按照对区间的处理方向, 可划分为分割(splitting), 合并(merging) 与混合(hybrid) 方法. 这里所谓的方向, 是将各层次的区间看作一个类似于决策树模型的树状结构, 顶端根节点为包含了全部数值的单一区间, 每一上层区间可分割出更多置于其下层的小区间, 随着从根节点向下移动, 各层的区间数量逐渐增多. 若离散化自树状结构的上层向下层移动, 这称为自上而下(top-down). 它对应着分割法, 即一般先将变量的所有取值归入一个区间, 然后逐步加入一些分割点, 划分出更多的区间, 直至满足一定的停止条件. 若离散化是沿着树状结构的下层向上层移动, 称为自下而上(bottom-up). 它对应合并法, 即先将每一数值单独作为一个区间, 然后不断合并相邻区间, 直至满足一定的停止条件. 混合法是分割与合并的混合, 一般分为两步: 第一步分割得到若干区间; 第二步合并某些区间, 以改进第一步产生的结果. 显然, 当数据量较大、数据取值较多时, 分割法更为高效, 许多决策树模型(如CART模型) 中的离散化采用分割法.

(5) 按照是否设置参数值, 可划分为参数离散化和非参数离散化. 参数离散化方法需要输入设置的参数(如最大的区间数量). 非参数离散化方法不需要人为设置参数, 可根据数据信息自动实现离散化.

(6) 按照离散化适用的群体范围, 可划分为全局(global) 与局部(local) 离散化. 全局离散化的结果适用于全体样本观测. 局部离散化则允许一个变量在不同的群体中实施有区别的离散化. 例如, 在决策树CART模型中, 某一变量在两个中间节点都作为分割变量, 但分割的临界值不同, 从而在两个群体中形成了不同的离散化.

表 6.1: 几种离散化方法的分类

离散化方法	离散化方法分类方式[1]					
	1	2	3	4	5	6
组距分组	无监督	静态	直接	分割	参数	全局
分位数分组	无监督	静态	直接	分割	参数	全局
均值-标准差分组	无监督	静态	直接	分割	参数	全局
K-means分箱法	无监督	静态	直接	分割	参数	全局
ChiMerge算法	有监督	静态	递增	合并	非参数	全局
CAIM算法	有监督	静态	递增	分割	非参数	全局
基于MDLP的离散化方法	有监督	静态	递增	分割	非参数	全局
CART模型的离散化方法	有监督	动态	递增	分割	非参数	局部

[1] 1-6分别对应文中离散化方法分类方式(1)-(6).

表6.1列出了本章将要介绍的几种离散化方法所属类别. 作为对比, 也列出了CART模型的离散化方法所属类别.

6.2 无监督离散化

无监督离散化方法不使用后续数据分析中模型因变量的信息, 仅使用自变量本身的信息进行离散化. 下面, 介绍四个无监督离散化方法: 组距分组、分位数分组、均值-标准差分组、K-means分箱法.

6.2.1 组距分组

组距分组(width discretization) 按照设置的组距将数据分为若干个区间, 包括等距(equal width) 和不等距(unequal width) 分组. 组距的确定可以根据实际业务的需求.

对于等距分组, 可以先确定组数$k(k \geq 2)$, 然后测算出组距$w = (x_{\max} - x_{\min})/k$, 其中$x_{\max}$和$x_{\min}$分别是数据的最大值和最小值. 由此获得$k-1$个分割点$x_{\min} + w, x_{\min} + 2w, \cdots, x_{\min} + (k-1)w$.

对于不等距分组, 由于组距的影响, 各组频数不可比. 可以先消除组距的影响, 再进行组间对比. 例如, 可使用频数密度, 它等于频数除以组距.

直方图是应用组距分组的一个典型. 需要注意, 组数可能会影响变量的分布形态, 组数不应太少也不能太多, 以保留足够多的分布信息, 同时又起到直

观简洁展示的作用. 组数的确定可以参考Sturges经验公式:

$$k = 1 + 3.3\lg(n),$$

其中n表示样本量.

6.2.2 分位数分组

分位数分组(frequency discretization) 以某些分位数为分割点将数据分为若干个区间, 包括等频(equal frequency) 和不等频(unequal frequency) 分组.

对于等频分组, 可以先确定组数$k(k \geq 2)$, 则每一组中的样本量约为n/k, 其中n为样本量. 但由于相同的值必须放入同一个区间, 因此最终不一定能保证每一组的频数或频率都相等.

相对于组距分组, 分位数分组可以保证各组的频数或频率大小. 需要注意的是, 当采用样本分位数进行分组时, 分组效果会受到样本分位数估计效果的影响.

6.2.3 均值-标准差分组

均值-标准差分组以均值μ为中心, 往均值的两边各走若干倍标准差σ的长度, 将这些位置作为分割点, 获得若干个组. 例如, 以$\mu \pm \sigma$作为分割点, 可以将变量取值分为三个组, 即$(-\infty, \mu - \sigma], (\mu - \sigma, \mu + \sigma]$和$(\mu + \sigma, \infty)$. 它们体现了数据值偏离中心的距离和方向.

6.2.4 K-means分箱

K-means分箱方法采用K-means聚类方法将变量取值聚为若干个组. 实际上, 除了K-means聚类方法, 其他适用于单变量聚类的方法都可用于离散化.

6.3 有监督离散化概述

有监督离散化需要使用后续数据数据分析中模型的因变量Y. 一般要求Y为定性变量, 离散化的主要目的是更高效地预测Y的类别.

6.3.1 有监督离散化的基本原则

直观上看, 有监督离散化需满足四条基本原则:

(1) 当按照离散化后的变量对数据分组时, 各个组在因变量Y上应能表现出明显的趋势特征. 这表明离散化后的变量与Y具有相关性, 即具有一定的预测能力.

(2) 在离散化后变量的相邻区间上, Y的分布差异应该较大. 这是为了避免不必要的区间分割, 从而尽量减少离散化后变量的取值数量.

(3) 离散化后变量的取值数量应当适中. 取值数量过少导致变量的区分度不足, 过多则不便于管理且可能导致模型的稳定性不强.

(4) 各区间内样本数量合理, 不应过多或过少.

由此, 可以将有监督离散化看作一个优化问题. 根据原则(1), 将有监督离散化的优化目标设置为：离散化后的变量尽可能大地提升对Y的分类预测效果. 根据(2)-(4), 可以为优化目标设置约束条件. 但适中的区间数量和区间内样本数量, 会因问题和数据不同而具有很大差异性, 无法统一界定. 为便于数学上的处理, 我们将约束条件简化为: 离散化后的变量取值尽量少. 从而, 有监督离散化转化为一个带约束的优化问题.

6.3.2 有监督离散化效果评价

常用两类指标评价离散化效果.

第一类评价指标为统计度量(statistical measure), 一般是离散化后的变量与因变量的相关度. 相关度越大, 表明离散化后的变量对Y的分类预测效果越好, 离散化的效果也就越好. 6.4和6.5节中的ChiMerge算法和CAIM算法即采用相关度作为评价指标.

第二类评价指标为信息量(information), 用于刻画变量的变异性. 信息量越小, 表明变量的变异性越小. 因此, 按照离散化后的变量对样本观测分组, 每一组内因变量Y的信息量越小, 表明离散化的效果越好. 将离散化前后的信息量之差定义为信息增益(information gain), 则信息增益越大, 表明离散化的效果越好. 6.6节中基于MDLP的离散化方法即采用了信息增益作为评价指标.

当因变量Y只含有两个类别时，有专门的离散化效果评价指标. 例如，评分卡模型中的因变量包含两个类别: “好客户” (未违约客户) 和“坏客户”(违约客户), 可采用证据权重(Weight of Evidence, WOE) 辅助评价离散化效果. 对每个离散化后的区间, WOE的计算公式为：

$$\mathrm{WOE} = \ln\left(\frac{\text{好客户占比}}{\text{坏客户占比}}\right) \times 100.$$

例6.1. 假设有3000位客户, 一半是好客户, 一半是坏客户[1]. 对收入进行离散化, 划分为3个区间. 表6.2是收入类别与是否违约的交叉表. 由此, 可以计算每一个区间的WOE值. 例如, 在收入小于1000元的客户群体中, 未违约的客户占全体1500位未违约客户的31.1%, 违约的客户占全体1500位违约客户的16.1%, 因此WOE值为

$$\mathrm{WOE}_1 = \ln\left(\frac{0.311}{0.161}\right) \times 100 \approx 65.94.$$

可类似获得其余区间的WOE值, 列于表6.2中. 可以看到, 收入小于1000元的客户更倾向于不违约, WOE值较大; 收入在1000至2400元的客户更倾向于违约, WOE值为负值, 且绝对值较大; 而收入大于等于2400元的客户违约与不违约的比例相当, WOE值较小.

由例6.1可见, 我们可以通过WOE的绝对值大致评价经离散化处理的变量各取值对于预测的贡献. WOE的绝对值越大, 表明对应取值的群体在因变量上呈现了更为明显的特征, 预测的准确率可能更高.

进一步汇总各WOE值, 形成对离散化变量的整体性评价, 这就是信息值(Information Value, IV), 其计算公式为:

$$\mathrm{IV} = \sum_{i=1}^{k} (\text{好客户占比}_i - \text{坏客户占比}_i) \times \mathrm{WOE}_i, \tag{6.1}$$

其中k为经离散化处理的变量取值个数.

可以将IV看作WOE的加权平均, 所采用的"权重"是好客户占比与坏客户占比之差. 这一"权重"很特殊, 其取值可能大于0, 也可能小于0. "权重"的正负性与WOE的正负性恰好一致, 二者相乘大于等于0. 这就相当于对WOE取了绝对值, 从而按照WOE的绝对值大小衡量离散化后变量的预测性能. 此外, 由于离散化后的变量取某一值的比例越大, 该值对应的好客户占比和坏客户占比也可能越大, 因此该"权重"在一定程度上反映了离散化后变量各取值的规模, 从而将规模效应纳入评价之中.

IV可用于评价定性自变量对预测因变量Y的效果. IV越大, 表明自变量对预测的贡献越大. 经验上的判断规则为:

(1) 当$\mathrm{IV} < 2$时, 自变量对预测因变量几乎无帮助;

(2) 当$2 \le \mathrm{IV} < 10$时, 自变量对预测因变量有一定帮助;

[1]数据来源于张文彤, 钟云飞(2013). 该数据中好客户与坏客户比例不太符合实际情况, 现实中好客户所占比例往往远高于坏客户比例. 可以把这个数据看作是经过类别不平衡处理后的数据.

表 6.2: 计算WOE和IV的示例

收入		是否违约		合计	WOE_i	IV_i
		未违约	违约			
< 1000	计数	466	241	707	65.94	9.89
	列百分比	31.1%	16.1%	23.6%		
[1000, 2400)	计数	411	691	1102	-51.95	9.71
	列百分比	27.4%	46.1%	23.6%		
≥ 2400	计数	623	568	1191	9.24	0.33
	列百分比	41.5%	37.9%	39.7%		
合计	计数	1500	1500	3000		19.93
	列百分比	100.0%	100.0%	100.0%		

(3) 当$10 \le \text{IV} < 30$时, 自变量对预测因变量有较大帮助;

(4) 当$\text{IV} \ge 30$时, 自变量对预测因变量有很大帮助. 但是, 当$\text{IV} > 50$时, 自变量对因变量有过度预测的倾向.

需要注意的是, 当IV特别大时, 我们不应盲目乐观. 例如, 若以由Y衍生的变量作为自变量, 其IV往往较大. 但当模型应用于预测场景时, 由Y衍生的变量一般不能在预测之前获得, 因而不能作为模型的自变量.

例6.1 (续) 计算收入类别的IV. 为便于展示, 我们将(6.1)式求和项中的第i项记作IV_i $(i = 1, \cdots, k)$. 例如, 对于第一个区间,

$$\text{IV}_1 = (0.311 - 0.161) \times 65.94 \approx 9.89.$$

类似可获得其他IV_i, 结果列于表6.2中. 汇总IV_i $(i = 1, \cdots, k)$, 得到IV= 19.93, 它大于10且小于30, 表明离散化后的变量对预测因变量有较大帮助.

6.4 ChiMerge算法

ChiMerge算法由Kerber (1992) 提出, 以相关度最大为优化目标, 是针对单一变量、自下而上的有监督离散化方法. ChiMerge算法的主要思想是：先对数据排序并将每一数值单独作为一个区间, 自下而上逐步合并无显著差异的相邻区间, 其中区间差异性通过列联表检验(contingecy table test)[2] 予以识别. 可见, "ChiMerge"这一名称体现了该算法的特点, 即采用列联表检验判断相邻区间差异性, 并且采用合并区间的方式.

[2]列联表检验基于卡方检验(Chi-square test) 实现.

为理解ChiMerge算法, 先简要介绍列联表检验. 列联表检验用于检验两个定性变量的相关性, 它将假定变量独立的情况与数据真实的情况进行对比, 二者的差异性由卡方统计量予以衡量. 卡方统计量越大, 表明两种情况差异越大, 从而认为两个定性变量越有可能不独立、具有相关性. 在一定的条件下, 卡方统计量近似服从卡方分布$\chi^2((a-1)\times(b-1))$, 其中$(a-1)\times(b-1)$为自由度, a和b分别为两个定性变量的取值个数.

在ChiMerge算法中, 为考察两个相邻区间中因变量Y的分布是否有显著差异, 应构造一个变量A, 它只有两个取值, 每一取值对应于一个区间. 将A与Y作列联表检验, 根据列联表检验的原理, 由于A的取值个数为2, 卡方统计量近似服从$\chi^2(k-1)$, 其中k为Y的取值个数. 若卡方统计量显著地大, 则A与Y的相关性较强, 表明Y在两个相邻区间上分布的差异较大, 不应合并区间; 反之, 若卡方统计量并不显著地大, 表明Y在两个区间上分布的差异较小, 则可以合并区间. 为判断卡方统计量是否"显著地大", 需设置置信水平$1-\alpha$(一般取$0.5 < 1-\alpha < 1$), 找到卡方分布在该置信水平下的分位数$\chi^2_{1-\alpha}(k-1)$. 若卡方统计量大于$\chi^2_{1-\alpha}(k-1)$, 则不应合并区间; 若卡方统计量小于等于$\chi^2_{1-\alpha}(k-1)$, 则应该合并区间. 若有多对相邻区间的卡方统计量小于等于$\chi^2_{1-\alpha}(k-1)$, 应优先合并卡方统计量最小者.

由此, ChiMerge算法的具体步骤是:

步骤1: 将待离散化的变量取值排序, 将每一个取值作为一个区间. 也可以通过无监督离散化等方式产生若干初始区间.

步骤2: 将每一对相邻区间与因变量Y构造为列联表, 获得卡方统计量的值.

步骤3: 合并步骤2中卡方值最小且小于临界值$\chi^2_{1-\alpha}(k-1)$的一对相邻区间.

步骤4: 重复步骤2, 直到仅有一个区间或所有相邻区间的卡方值都大于临界值$\chi^2_{1-\alpha}(k-1)$.

例6.2. 采用golf数据集中的4个样本观测(如表6.3所示), 因变量为二分类的定性变量Play, 对定量变量Humidity进行离散化. 由于因变量Play的取值个数为2, 所以卡方分布的自由度为2-1=1. 取置信水平为0.9, 可以查到自由度为1的卡方分布的0.9分位数$\chi^2_{0.9}(1) = 2.71$. 下面, 应用ChiMerge算法, 首先执行第一轮:

步骤1: 按照Humidity取值从小到大将数据排序, 将每一个取值作为一个区间. 这里, 我们取相邻数值的中间值为分割点, 形成了四个初始区间(如表6.4"第一轮"所示).

表 6.3: golf数据集中的4个样本观测

序号	Humidity	Play
1	85	No
2	90	No
3	86	Yes
4	96	Yes

表 6.4: ChiMerge算法过程

第一轮			第二轮			第三轮		
Humidity	Play		Humidity	Play		Humidity	Play	
	No	Yes		No	Yes		No	Yes
$80 \rightarrow [80, 83)$	0	1	$80, 86 \rightarrow [80, 88)$	0	2	$80, 86 \rightarrow [80, 88)$	0	2
$86 \rightarrow [83, 88)$	0	1						
$90 \rightarrow [88, 93)$	1	0	$90 \rightarrow [88, 93)$	1	0	$90, 96 \rightarrow [88, 96]$	1	0
$96 \rightarrow [93, 96]$	0	1	$96 \rightarrow [93, 96]$	0	1			

步骤2: 将每一对相邻区间与因变量Play构造为列联表. 例如, 将前两个区间与Play构造为2行2列的列联表, 得到卡方统计量的值为0. 类似地, 将第二和第三个区间与Play构造列联表, 卡方统计量的值为2; 将第三和第四个区间与Play构造列联表, 卡方统计量的值也为2.

步骤3: 合并步骤2中卡方值最小且小于临界值的一对相邻区间. 可以看到, 前两个区间对应的卡方统计量值最小, 并且小于临界值2.71, 因此应合并第一与第二个区间.

通过第一轮离散化, 将Humidity取值变换为三个区间, 它们是第二轮离散化的初始情况, 列于表6.4“第二轮”中. 类似地构造各对相邻区间与因变量Play的列联表, 得到卡方统计量的值分别为3和2. 第二对相邻区间对应的卡方统计量值最小, 并且小于临界值2.71, 因此应合并第二对相邻区间.

在第三轮离散化中(如表6.4“第三轮”所示), 只需要进行一次列联表检验, 得到卡方统计量的值为1.33, 小于临界值2.71, 因此应合并区间. 从而, 将Humidity取值变换为一个区间. 此时, 达到停止条件, ChiMerge算法完成.

在例6.2中, 离散化后的Humidity只有一个取值. 这样的结果可以为我们提供一个额外的信息, 那就是Humidity对因变量不具有较高的预测能力. 在6.1节中提到, 离散化具有一定的变量筛选功能, 实际上就体现在这样的结果中. 若离散化后的变量只有一个取值, 说明其对应的原始自变量对因变量的预测效果不佳, 可以考虑剔除该自变量, 从而提升建模效率.

最后, 讨论ChiMerge算法存在的几个问题.

第一个问题是, ChiMerge算法所获得的列联表检验结果可能不精确. 一方面, ChiMerge算法所采用的卡方分布自由度并不精确, 因为在自变量的某对相邻区间内, 因变量可能只取少数的类别值[3], 此时自由度应取更小的值. 也就是说, ChiMerge算法所取的卡方分布自由度为所有可能自由度中的最大值. 在相同的置信水平下, 自由度越大, 卡方分布的分位数值越大. 过大的临界值会导致合并更多的区间, 从而造成过度合并的(overmerged) 问题. 另一方面, 满足自变量取某一值的样本观测可能较少, 导致列联表中的频数较小、不符合卡方统计量渐近服从卡方分布所需要的假设, 影响离散化的效果.

第二个问题是, ChiMerge算法需要人为设置置信水平, 使得离散化结果受到一定的主观性影响.

第三个问题是, 当数据量较大、数据取值较多时, ChiMerge这类自下而上的方法较为低效.

[3]若待离散化的自变量与因变量具有较强的关联, 这种情况可能较为普遍.

表 6.5: CAIM算法中的数据汇总表

Y	D					行和
	$[d_0, d_1]$	$\cdots$	$(d_{r-1}, d_r]$	$\cdots$	$(d_{n-1}, d_n]$	
C_1	q_{11}	$\cdots$	q_{1r}	$\cdots$	q_{1n}	M_{1+}
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
C_i	q_{i1}	$\cdots$	q_{ir}	$\cdots$	q_{in}	M_{i+}
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
C_k	q_{k1}	$\cdots$	q_{kr}	$\cdots$	q_{kn}	M_{k+}
列和	M_{+1}	$\cdots$	M_{+r}	$\cdots$	M_{+n}	M

6.5 CAIM算法

CAIM算法由Kurgan & Cios (2004) 提出. 与ChiMerge算法类似的是, CAIM算法针对单一变量进行离散化, 并且以离散化后的变量与因变量的相关度作为优化目标. 但与ChiMerge算法不同的是, CAIM算法采用CAIM (Class-Attribute Interdependency Maximization) 作为相关度指标. 此外, CAIM算法采用自上而下的思路, 即先将所有数据点归入一个或若干个区间, 然后逐级动态加入分割点, 划分出更多的区间. 具体而言, CAIM算法考虑了三个方面: (1) 最大化离散化后的变量与因变量的相关度; (2) 最小化区间数量; (3) 计算开销合理, 可适用于有大量取值的变量.

首先介绍一些记号. 设因变量Y有k个类别, 取值为$C_1, C_2, \cdots, C_k$. 设待离散化的变量为F, 将F离散化后的变量为D, D包含n个区间$[d_0, d_1], \cdots, (d_{n-1}, d_n]$. CAIM算法将离散化后的变量$D$与因变量$Y$构造为列联表(如表6.5所示), 每一行对应$Y$的一个类别, 每一列对应$D$的一个取值. 以$q_{ij}$表示满足$Y = C_i$且$D = (d_{j-1}, d_j]$的样本数量. 将第$i$行的所有$q_{ij}$相加, 得到行和$M_{i+}$, 将第$j$列的所有$q_{ij}$相加, 得到列和$M_{+j}$. 总的样本量为$M = \sum_{i=1}^{k} M_{i+} = \sum_{j=1}^{n} M_{+j}$.

基于上述记号, CAIM算法提出CAIM指标, 以度量因变量Y与离散化变量D的相关性. CAIM的表达式为:

$$\mathrm{CAIM}(Y, D|F) = \frac{\sum_{r=1}^{n} \frac{\max_r^2}{M_{+r}}}{n},$$

其中n为区间个数, $\max_r$为第r列中各q_{ir} $(i = 1, \cdots, k)$的最大值.

可以看到, CAIM越大, 因变量与离散化后的变量相关性越高. CAIM指标的构造主要基于以下考虑:

表 6.6: 用于计算CAIM指标的数据汇总表

以83为分割点			以88为分割点			以93为分割点		
Play	[80, 83)	[83, 96]	Play	[80, 88)	[88, 96]	Play	[80, 93)	[93, 96]
No	0	1	No	0	1	No	1	0
Yes	1	2	Yes	1	2	Yes	2	1
$\max_r$	1	2	$\max_r$	2	1	$\max_r$	2	1
M_{+r}	1	3	M_{+r}	2	2	M_{+r}	3	1

(1) $\max_r$越大, 表明落入区间中的样本观测在Y上的取值越集中, 从而Y与D的相关性越强.

(2) 纳入M_{+r}是为了考虑未被分入多数类的样本观测所产生的负面影响.

(3) 由于希望区间数量尽可能小, 因此将区间数量n放置在分母, 相当于对较大的区间数量予以惩罚.

CAIM算法采用贪心搜索, 以获得局部最大的CAIM指标. 算法主要包含两个步骤:

步骤1: 初始化区间, 并计算CAIM指标. 允许初始化区间为多个区间.

步骤2: 加入一个使得CAIM指标达到最大的分割点.

重复步骤2, 直到CAIM指标不再增大且区间数大于等于因变量的类别数.

例6.3. 采用例6.2中的数据, 对定量变量Humidity进行离散化. Humidity有4个取值, 排序后为80、86、90、96. 下面, 执行CAIM算法.

步骤1: 初始化区间. 本例中, 将初始区间设置为一个, 即[80,96]. 备选的分割点为83、88和93. 此时, $max_r = 3, M_{+r} = 4, n = 1$, 因此CAIM= 9/4.

步骤2: 为找到一个使得CAIM指标达到最大的分割点, 需要考察每一个备选分割点. 表6.6列出了相应的数据汇总表, 由此分别得到以83、88和93为分割点所对应的CAIM指标值, 为7/6、5/4和7/6. 其中最大的CAIM指标值为5/4. 由于5/4小于分割前的CAIM指标值9/4, 因

此从CAIM指标来看, 不需要再加入分割点, 应保持离散化变量取值为一个区间. 但是, CAIM算法强制要求离散化变量的取值数量不小于因变量的类别数, 因此这里仍然需要分割. 从而, 取使得CAIM指标达到最大的分割点88, 产生$[80, 88)$和$[88, 96]$两个区间.

按照CAIM算法, 应重复步骤2, 直到满足停止条件. 留作习题.

6.6 基于MDLP的离散化方法

基于MDLP的离散化方法由Fayyad & Irani (1993) 提出. 它基于信息增益寻找最优的分割点, 并且采用自上而下的方式, 逐级、动态地分割区间. 这类似于依据单一自变量构建决策树CART模型的方式. 与CART模型的构建所不同的是, 基于MDLP的离散化方法需要对信息增益最大的分割点作进一步判断, 其判断准则基于最小描述长度原则(Minimum Description Length Principle, MDLP). 因此, 该方法被称为基于MDLP的离散化方法.

6.6.1 基于信息增益的离散化

基于MDLP的离散化方法以熵(entropy) 作为信息量, 从而以信息增益最大化作为优化目标.

信息量也称为不纯度(impurity). 熵是一种常用的不纯度指标, 用于刻画定性变量的波动大小. 熵这一概念来自物理学, 用于描述系统是否处于平衡状态, 熵越大表明系统越平衡, 也就是各个类别"势力"相当, 从而表现出越为混沌的状态. 之后, 信息学领域采用熵这一名称, 定义了用于衡量变量信息长度的熵, 其表达式为:

$$Ent = -\sum_{i=1}^{k} p_i \log_2 p_i,$$

其中k是变量的取值个数, p_i是变量取第i个值的概率$(i = 1, \cdots, k)$. 当$p = 0$时, 定义$p \log_2 p = 0$. 由香农定理, 应以$-\log_2 p_i$作为第i条信息的长度[4]. 直观来看, 信息出现的概率越大, 那么赋予该信息的长度就越小, 以便于提升信息传送效率. 熵是将信息长度乘以其概率并求和, 表达了信息长度的期望值. 熵的取值范围为$[0, \log_2 k]$. 当熵为最小值0时, 变量只取1个值, 达到最"纯"的状态; 当熵为最大值$\log_2 k$时, 变量取各值的概率都为$1/k$, 处于最平衡、最"不纯"的状态. 我们希望离散化后的区间内因变量是"纯"的, 因此离散化后的熵越小越好.

对数据集S, 记因变量的熵为$Ent(S)$. 按照变量A的一个分割点T, 将S分割为两个互不重叠的子集S_1和S_2, 将两个子集的熵按样本量进行加权平均,

[4] 当取$\log_2$时, 可以将信息长度看作以bit为单位.

可以获得区间分割后的熵, 为:

$$\frac{|S_1|Ent(S_1)+|S_2|Ent(S_2)}{N},$$

其中N表示数据集S的样本量, $|S_1|$和$|S_2|$分别表示数据集S_1和S_2的样本量.

信息增益用于衡量区间分割所造成的增益量, 计算方式为区间分割前的不纯度减去分割后的不纯度. 若以熵作为不纯度的刻画, 信息增益为:

$$Gain = Ent(S) - \frac{|S_1|Ent(S_1)+|S_2|Ent(S_2)}{N}.$$

将区间分割前的不纯度作为固定值, 我们希望离散化后的熵越小越好, 也即希望信息增益越大越好.

在基于MDLP的离散化方法中, 所选取的分割点需要满足一定的条件:

(1) 区间分割所获得的信息增益应大于0, 否则分割无效.

(2) 区间分割是最优的. 这表明, 选取的分割点所获得的信息增益大于等于其他候选分割点所获得的信息增益.

(3) 满足MDLPC准则中的接受分割点条件.

条件(1)和(2)与大部分决策树算法类似. 这里的关键是理解条件(3), 下面作具体介绍.

6.6.2 MDLPC准则

定义6.1. (MDLPC准则) 设有样本量为N的数据集S, 对于待离散化变量A的某一分割点T, 当且仅当

$$Gain(A,T;S) > \frac{\log_2(N-1)}{N} + \frac{\Delta(A,T;S)}{N} \tag{6.2}$$

时, 接受以分割点T进行区间分割; 否则, 拒绝以分割点T进行区间分割. 其中,

$$\Delta(A,T;S) = \log_2(3^k-2) - (kEnt(S) - k_1Ent(S_1) - k_2Ent(S_2)),$$

k, k_1, k_2分别为数据集S, S_1, S_2中因变量的取值个数, 满足$k \geq 1$, $1 \leq k_1, k_2 \leq k$.

要理解和推导MDLPC准则, 可以借助于对信息传送的描述. 假设信息发送者掌握了全部信息, 而信息接收者仅掌握了全体样本观测的自变量信息, 但缺乏因变量Y的信息. 发送者需要以最小的描述长度将Y的信息传送给接收者. 以$Ent(S)$表示数据集S中基于Y计算的熵, 它刻画了Y各取值的平均

描述长度. 由于发送者需要传送每一个样本观测的Y, 当有N个样本观测时, 总共需要的描述长度为$NEnt(S)$. 此外, 由于发送的是Y编码后的信息, 发送者还需要传送一个代码手册, 告诉接收者Y的编码与其原始值的对应关系. 若Y的取值个数为k, 那么传递代码手册信息的描述长度为$kEnt(S)$. 因此, 若拒绝分割点T, 那么按照一个数据集S进行信息传送所需要的成本, 即总的描述长度为$NEnt(S) + kEnt(S)$, 记为$Cost(NT)$.

如果接受分割点T, 总的描述长度会发生什么变化呢? 我们假设变量A在数据集S中有N个取值, 那么候选的分割点共有$N-1$个. 由分割点T将数据集S划分为S_1和S_2两个互不重叠的部分, S_1和S_2的样本量分别为$|S_1|$和$|S_2|$. 接受分割点T的总描述长度由三个部分构成:

(1) 由于待离散化变量A有$N-1$个候选的分割点, 从而分割点T的描述长度等于$\log_2(N-1)$.

(2) 传送S_1和S_2中每一个样本观测的Y, 描述长度为$|S_1|Ent(S_1)$+$|S_2|Ent(S_2)$.

(3) 传送代码手册的描述长度为$\log_2(3^k-2) + k_1Ent(S_1) + k_2Ent(S_2)$, 其中$k, k_1, k_2$分别为数据集$S, S_1, S_2$中因变量的取值个数.

在传送代码手册的描述长度中, $k_1Ent(S_1)$和$k_2Ent(S_2)$分别表示S_1和S_2中Y的编码与其原始值对应关系的描述长度. 另一项$\log_2(3^k-2)$需要着重解释, 它来源于发送者需要告诉接收者, S中Y的取值分配到S_1和S_2中的情况. 当$0 < k_1 < k$时, 那么Y的其余$k-k_1$个值都必须包含于S_2中; 若假定$|S_2| - (k-k_1) \geq k_1$, S_2还可以含有S_1中Y的任意多个值, 这样的组合数为2^{k_1}. 当$k_1 = k$时, 由于S_2不能为空集, 所以S_2中Y的取值情况共有2^k-1种, 即在2^k种组合的基础上排除S_2不包含Y的任一取值的情况. 因此, 总共有$\Sigma_{k_1=1}^{k} C_k^{k_1} 2^{k1} - 1 = 3^k - 2$种可能. 还可以从另外一个角度分析, Y的某一取值可以只包含于S_1, 或只包含于S_2, 或同时包含于S_1和S_2, 因此共有3^k种可能. 但由于S_1和S_2都不能为空集, 所以要排除Y的所有取值都未包含于S_1和Y的所有取值都未包含于S_2这两种情况, 因此总共是3^k-2种可能. 将这些可能的情况传送给接收者, 需要的描述长度为$\log_2(3^k-2)$.

若接受分割点T, 那么传送成本, 即总的描述长度$Cost(HT)$就等于以上三部分描述长度之和. 当接受分割点T所产生的成本更低时, 即当$Cost(HT) < Cost(NT)$时, 才能接受分割点T. 经过简单的数学推导, 这一条件等价于(6.2)式.

6.6.3 方法步骤

基于MDLP的离散化方法具体为: 首先初始化, 形成若干个新区间. 例如, 将所有数值合并为一个新区间; 然后执行以下两个步骤:

表 6.7: 按Humidity从小到大排序后的数据

Play	yes	yes	no	yes
Humidity	80	86	90	96

步骤1: 针对每一个新区间: 对区间内所有备选的分割点, 计算信息增益值; 然后, 取所有信息增益值的最大值, 记为$\max Gain$. 若$\max Gain < 0$, 不产生任何新区间. 若$\max Gain > 0$, 则采用MDLPC准则判断是否可以接受$\max Gain$对应的分割点, 若可以接受, 则按该分割点产生两个新区间; 否则, 不产生新区间.

步骤2: 若步骤1中产生了新区间, 对每一新区间执行步骤1. 直到不再产生新区间.

Fayyad & Irani (1993) 指出, 分割点T的搜索范围可缩小到待离散化变量取值的一个子集之中. 为此, 先介绍一个定义.

定义6.2. 将集合V中的数值排序为$v_1, v_2, \cdots$. 若存在某个i使得v_i与v_{i+1}对应因变量的不同类别, 则任一满足$v_i < T < v_{i+1}$的数值T称为V的**边界点(boundary point)**.

下面的定理告诉我们, 只需要在边界点中搜索分割点, 以提升搜索效率.

定理1. 若分割点T使得离散化后的信息增益达到最大, 则T一定是待离散化变量的边界点.

例6.4. 采用例6.2中的数据, 对定量变量Humidity进行离散化. 首先进行初始化, 将所有数值合并为1个区间, 即[80,96]. 初始化区间的熵为:

$$-\frac{3}{4}\log_2\frac{3}{4}-\frac{1}{4}\log_2\frac{1}{4}\approx 0.81.$$

按照Humidity从小到大排序(如表6.7所示), 其中88和93为边界点.

下面, 执行步骤1. 对于新区间$[80, 96]$, 分别计算候选分割点所产生的信息增益. 以88为分割点所产生的信息增益为:

$$Gain = 0.81 - (0.5 \times 0 + 0.5 \times 1) = 0.31,$$

以93作为分割点所产生的信息增益为:

$$Gain = 0.81 - (0.75 \times 0.92 + 0.25 \times 0) = 0.12.$$

因此以88作为分割点的信息增益达到最大, 且信息增益值大于0.

还需要检测MDLPC准则中的接受分割点条件. 以S_1表示Humidity小于88的样本观测构成的集合, S_2表示Humidity大于等于88的样本观测构成的集合, 从而$k = 2, k_1 = 1, k_2 = 2, Ent(S) = 0.81, Ent(S_1) = 0, Ent(S_2) = 1$. 由此得到

$$\frac{\log_2(N-1)}{N} + \frac{\Delta(A,T;S)}{N} \approx \frac{\log_2(4-1)}{4} + \frac{3.19}{4} \approx 1.19.$$

分割点88所产生的信息增益小于该值, 因此拒绝以88为分割点进行分割. 由于不再产生新区间，因此算法终止. 最终得到的离散化结果是将Humidity的值合并为一个区间$[80, 96]$.

通过这个例子可以看到, 基于MDLP的离散化方法通过MDLPC准则控制离散化后变量的取值数量, 从而实现了数值归约.

6.7 案例分析

基于car数据集, 划分训练集和测试集.

```
alldata = pd.read_csv('car.csv')

y = alldata[['Mileage']]
X = alldata[['Country', 'Reliability', 'Price', 'Type',
      'Weight', 'Disp.', 'HP']]
X_train, X_test, y_train, y_test = train_test_split(
    X,y, test_size=1/3, random_state=1)
```

下面, 基于训练集训练离散化方法, 然后应用到测试集中. 首先, 采用KBinsDiscretizer函数实现无监督离散化.

由于KBinsDiscretizer函数不允许缺失值, 因此先对训练集作预处理, 取出定量变量, 以中位数插补定量变量的缺失值. 预处理后的训练集为X_trainpreprocess.

```
numeric_features = [2,4,5,6]
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median'))])

preprocessor = ColumnTransformer(transformers=[('num',
    numeric_transformer, numeric_features)])
```

```
X_trainpreprocess = pd.DataFrame(preprocessor.fit_transform(
    X_train), columns=(X_train.columns[numeric_features].values
    ))
```

1. 等距分组

在KBinsDiscretizer函数中设置strategy='uniform', 可实现等距分组; 由参数n_bins可设置各变量的区间数量; 设置encode='ordinal'可获得序号形式的离散化结果.

```
ew = KBinsDiscretizer(strategy='uniform', n_bins=[2,3,3,3],
    encode='ordinal')
X_train_bin = ew.fit_transform(X_trainpreprocess)
```

2. 等频分组

在KBinsDiscretizer函数中设置strategy='quantile', 可实现等频分组.

```
ef = KBinsDiscretizer(n_bins=[2,3,3,3], encode='ordinal',
    strategy='quantile')
X_train_bin = ef.fit_transform(X_trainpreprocess)
```

3. K-means分箱

在KBinsDiscretizer函数中设置strategy='kmeans', 可实现K-means分箱.

```
km = KBinsDiscretizer(n_bins=[2,3,3,3], encode='ordinal',
    strategy='kmeans')
X_train_bin = km.fit_transform(X_trainpreprocess)
```

上述无监督离散化返回的结果X_train_bin是取值为$0,1,\cdots$的序号, 表示样本观测被分入哪一个组. 通过属性bin_edges_ 查看各组边界值, 对应于左闭右开的区间.

下面, 实现有监督离散化. 有监督离散化需要定性的因变量. 本例中, 按照油耗Mileage是否小于等于25, 在训练集中形成二分类的因变量y_train_group (取值为0和1).

```
y_train_group = pd.cut(y_train['Mileage'], bins=[-np.infty,25,
    np.infty],include_lowest=True, labels=[0,1])
```

1. ChiMerge算法

采用ChiMerge函数实现ChiMerge算法. ChiMerge函数不允许缺失值, 因此采用经预处理的训练集X_trainpreprocess. 取置信水平confidence_level为默认值0.9, 最大区间数max_intervals=3, 最小区间数min_intervals=1.

```
trans_cm = ChiMerge(max_intervals=3, min_intervals=1,
    output_dataframe=True)
X_train_cm = trans_cm.fit_transform(X_trainpreprocess,
    y_train_group)
```

可由属性boundaries_ 查看分箱边界值(为每一区间的上限). 所产生的区间为左开右闭(最后一个区间上限为inf, 因此右端也为开区间). transform函数输出区间形式的离散化结果.

2. CAIM算法

采用CAIMD函数可以实现CAIM算法.

```
caim = CAIMD()
X_train_caim = caim.fit_transform(X_train.iloc[:,
    numeric_features], y_train_group)
```

通过属性split_scheme可查看各区间的下限和上限, 所产生的是左闭右开区间(最后一个区间两端都为闭区间). 输出值为区间的序号.

3. 基于MDLP的离散化方法

采用MDLP函数可以实现基于MDLP的离散化方法. MDLP函数不允许缺失值, 因此采用经预处理的训练集X_trainpreprocess. 设置随机数种子random_state为1, 其他参数取默认值.

```
transformer = MDLP(random_state=1)
transformer.fit(X_trainpreprocess, y_train_group)
```

由属性cut_points_可查看分割点(为分割处前后两个数值的均值). 可由transform函数获得以区间序号为取值的离散化结果, 若采用cat2intervals函数则获得区间形式的取值.

由各离散化方法所得的结果如表6.8所示. 三种方法对变量Disp.和HP的分割相似, 但在Price和Weight两个变量上的区间数量有所差异. 本例中, 自下而上的离散化方法(ChiMerge算法) 倾向于产生更多的区间.

Table 6.8: 数据离散化代码运行结果

方法	Price	Weight	Disp.	HP
ChiMerge	9410, 11470	2560, 2885	133	108
CAIM	9410	2710	133	108
基于MDLP的离散化方法	9995	2902.5	133	109

Table 6.9: 基于ChiMerge算法离散化结果的IV值

Weight	Price	Disp.	HP
18.37	16.65	3.77	1.95

可由WOE_Encoder获得离散化效果的评价指标WOE和IV. 设result_cm为区间分割点, 由此计算WOE和IV:

```
trans_woe = WOE_Encoder(output_dataframe=True)
result_woe = trans_woe.fit_transform(result_cm, np.array(
    y_train_group))
```

由属性result_dict_可以查看区间划分, 以及WOE和IV值. 例如, 按照表6.8中ChiMerge算法的区间分割点, 得到IV值的排序结果表6.9所示. 可以看到, 区间数量多的变量倾向于达到更高的IV值.

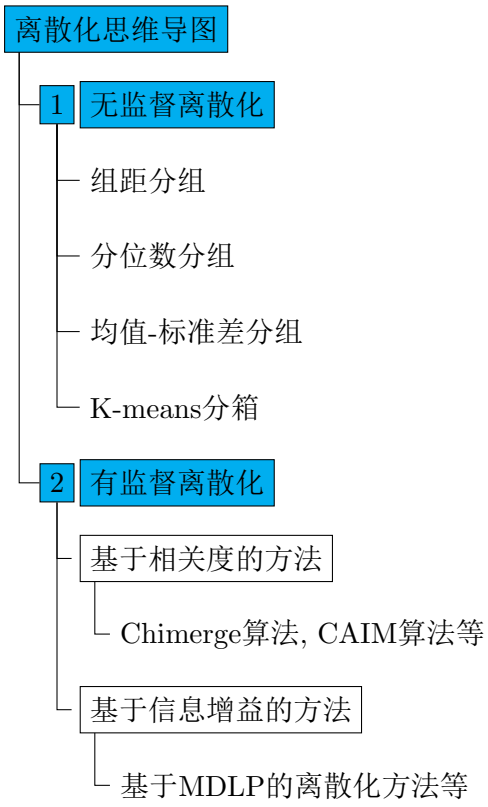
通过transform函数, 可将训练好的离散化方法应用于测试集. 例如, 将训练所得的K-means分箱应用于测试集(需要先按照训练好的数据预处理方法处理测试集):

```
X_testpreprocess = pd.DataFrame(preprocessor.transform(X_test)
    , columns=(X_train.columns[numeric_features].values))

X_test_bin = km.transform(X_testpreprocess)
```

本章小结

(1) 思维导图



(2) Python实现

表6.10列出了离散化的Python函数.

References

[1] Yeo I.K., Johnson R.A. (2000) A new family of power transformations to improve normality or symmetry. Biometrika, 87(4): 954-959.

[2] Box G.E.P., Cox D.R. An analysis of transformations. Journal of the Royal Statistical Society B, 26: 211-252.

[3] Kerber (1992). ChiMerge-discretization of numeric attributes. In National Conference on Artificial Intelligence, pages 123 – 128. AAAI Press.

表 6.10: 数据离散化Python函数

方法	函数名称与所属模块	重要参数	应用提示
无监督离散化	BinsDiscretizer (scikit-learn库preprocessing)	(1) strategy: 分组方式. 默认值为'quantile', 实现等频分组. 若设置为'uniform', 实现等宽分箱; 若设置为'kmeans' , 则实现基于Kmeans聚类的分组 (2) n_bins: 离散化后的组数, 默认值为5. 可取值为向量, 同时为多个变量设置组数(各变量的组数可以不同) (3) encode: 离散化后变量的类型. 默认值为'onehot', 以稀疏矩阵类型的独热编码变量表示分组结果. 若设置为'onehot-dense', 和'ordinal' , 则获得array数组类型的独热编码变量和以序号为取值的变量	(1) 函数不允许缺失值, 需处理缺失值 (2) 产生左闭右开的区间
ChiMerge算法	ChiMerge (scorecardbundle库 feature_discretization.ChiMerge)	(1) confidence_level: 置信水平, 默认值为0.9. 以置信水平对应的卡方分布分位数为临界值 (2) max_intervals: 最大区间数, 默认值为None (3) min_intervals: 最小区间数, 默认值为2 (4) initial_intervals: 初始化的区间数量, 通过分位数将数据等频分组. 这样的初始化可以提升离散化速度, 但离散化的结果可能并非最优. 默认值为100; 可设置为None, 对应每一个取值为一个区间 (5) output_dataframe: 输出结果的类型. 默认值为True, 输出结果为数据框类型. 若设置为False, 则输出结果为数组类型	区间范围为(-inf, inf); 产生左开右闭的区间(最后一个区间上限为inf, 因此右端也为开区间)
CAIM算法	CAIMD (caimcaim)	-	区间范围为训练集中的最小值到最大值, 测试集中的数据可能未落入该范围, 离散化结果为NaN; 产生左闭右开的区间(最后一个区间两端都为闭区间)
基于MDLP的离散化方法	MDLP (mdlp库discretization)	(1) continuous_features: 指定待离散化的定量变量. 默认值为None, 将所有变量作为定量变量, 分别进行离散化 (2) min_depth: 分割的最小深度, 默认值为0 (3) min_split: 可以产生区间分割的最小样本数量占总样本量的比例, 默认值为0.001 （4）random_state: 随机数种子. 当某些样本观测的自变量取值相同, 但因变量取值不同时, 需要对这些样本观测数据随机排序后确定分割点, 此时需要随机数种子. 默认值为None	(1) 函数不允许缺失值, 需处理缺失值 (2) MDLP函数中停止产生新区间的规则是满足以下条件中任一项: (i) max Gain小于等于0; (ii) 区间中的样本量小于等于min_split乘以总的样本量; (iii) 变量在区间中只有1个取值; (iv) 深度大于等于min_depth, 且根据MDLPC准则不接受区间中的最优分割点. 这一规则说明, 只有当深度大于等于min_depth时才可能触发基于MDLPC准则的判断 (3) 区间范围为(-inf, inf); 产生左开右闭的区间(最后一个区间上限为inf, 因此右端也为开区间)
有监督离散化效果评价(针对二分类因变量)	WOE_Encoder (scorecardbundle库 feature_encoding.WOE)	(1) epslon: 在除法运算和logrithm中加入的值, 以避免运算产生无穷值. 默认值为1e-10 (2) output_dataframe: 设置经过transform所产生的离散化数据是否为数据框. 默认值为False, 即产生的数据不是数据框, 而是numpy ndarray	基于区间分割点计算WOE和IV, 采用左开右闭的区间(最后一个区间上限为inf, 因此右端也为开区间)

[4] Kurgan L.A., Cios K.J. (2004) CAIM discretization algorithm. IEEE Transactions on Knowledge and Data Engineering, 16(2): 145-153.

[5] Fayyad U.M., Irani K.B. (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In Proceedings of the 13th International Joint Conference on Artificial Intelligence, pages 1022 – 1027.

[6] 张文彤, 钟云飞(2013) IBM SPSS数据分析与挖掘实战案例精粹. 北京: 清华大学出版社.