基于熵权法和支持向量机的中小微企业信贷决策研究

摘要

银行通常是根据中小微企业的实力、信誉对其信贷风险做出评估，再依据信贷风险等因素来确定是否放贷及贷款额度、利率和期限等信贷策略。本文以中小微企业的信贷数据为研究对象，对企业的信贷风险进行分析，为银行提供可靠的信贷调整策略。

针对问题一，本文建立基于主成分分析法的评价指标选择模型选出四个重要的评价指标。同时建立基于熵权法和支持向量机的信贷风险指数双重评价模型，得到每家企业的信贷风险指数。与信誉评级综合考虑选出可放贷的企业，并建立基于线性规划的信贷策略模型。经计算得到，对于附件 1 中 123 家企业，当**利率为 0.04** 时，信誉评分为 A、B、C 的客户流失率都为 0，此时年化利息最大，即此时的信贷策略是最优的。

针对问题二，本文基于问题一的模型，对附件二中的 302 家企业的信贷风险进行量化分析，得出这些企业的信誉等级和风险指数，并对贷款额度、贷款年利率等变量进行线性规划，得出该银行在年度信贷总额为 1 亿元、**利率为 0.0585** 时，信誉评分为 A、B、C 的客户流失率分别为 0.347、0.302、0.29，此时年化利息为 **408.16 万元**，即此时的信贷策略是最优的。

针对问题三，为了让信贷策略更加符合实际需求，本文通过查找相关文献数据，得到疫情期间各类企业受影响的波动值，利用这些波动值对各家企业的企业信息、进项发票与销项发票的数据进行预测更改，得到加入突发因素后的数据。再利用基于熵权法和支持向量机的信贷风险指数双重评价模型与基于线性规划的信贷策略模型得出该银行在年度信贷总额为 1 亿元时的信贷调整策略。

本文的主要**创新点**：

（1）通过主成分分析的方法，对企业信息的各种数据进行指标的选取，进而得到合理的客观的评价指标。

（2）建立基于熵权法和支持向量机的信贷风险指数双重评价模型能够更为可靠地评价企业的信贷风险，结合实际情况对问题进行求解，具有现实意义，且通用性较强。

（3）建立基于线性规划的信贷策略模型，依据银行的最大年化利息，得到准确的信贷策略。

关键词：主成分分析　熵权法　支持向量机 SVM　线性规划

一、问题重述

1.1 问题的背景

某银行对确定要放贷企业的贷款额度为10~100万元；年利率为 4%~15%；贷款期限为 1 年。在实际中，由于中小微企业规模相对较小，也缺少抵押资产，因此银行通常是依据信贷政策、企业的交易票据信息和上下游企业的影响力，向实力强、供求关系稳定的企业提供贷款，并可以对信誉高、信贷风险小的企业给予利率优惠。银行首先根据中小微企业的实力、信誉对其信贷风险做出评估，然后依据信贷风险等因素来确定是否放贷及贷款额度、利率和期限等信贷策略。

1.2 问题的叙述

根据题中的所给的信息，本文将题目中描述的 3 大问题建立 3 个数学模型，做出假设并建立数学模型进行求解。

问题1：对附件1中123家企业的信贷风险进行量化分析，给出该银行在年度信贷总额固定时对这些企业的信贷策略。

问题2：在问题1的基础上，对附件2中302家企业的信贷风险进行量化分析，并给出该银行在年度信贷总额为1亿元时对这些企业的信贷策略。

问题3：企业的生产经营和经济效益可能会受到一些突发因素的影响，而且突发因素往往对不同行业、不同类别的企业会有不同的影响。对附件2中的企业信息加入突发因素后，得到更加符合实际的企业信息。接着，利用问题一与问题二的模型，求解得出该银行在年度信贷总额为1亿元时的信贷调整策略。

二、问题分析

针对问题一，某银行对确定要放贷企业的贷款额度为 10~100 万元；年利率为 4%~15%；贷款期限为 1 年。对于附件 1 中 123 家企业，该银行在年度信贷总额固定时需要对这些企业的信贷风险进行量化分析，从而得到信贷策略。由于量化分析的指标较多，容易产生偏差，因此，指标权重的确定是一项重要的工作。权重的确定方法有很多，大致可分为主观赋权法、客观赋权法和组合赋权法。其中，熵权法是目前常用的一种客

观赋权法，其根据样本数据信息确定指标权重，避免主观性倾向的影响。

针对问题二，由问题一构建出基于熵权法和支持向量机的信贷风险指数双重评价模型对附件二中的302家企业的信贷风险进行量化分析，同时得出这些企业的信誉等级和风险指数，并对贷款额度、贷款年利率等变量进行线性规划，得出该银行在年度信贷总额为1亿元时对这些企业的信贷策略。

针对问题三，为了让信贷策略更加符合实际需求，本文通过查找相关文献数据，得到疫情期间各类企业受影响的波动值，利用这些波动值对各家企业的企业信息、进项发票与销项发票的数据进行预测更改，得到加入突发因素后的数据。再利用基于熵权法和支持向量机的信贷风险指数双重评价模型与基于线性规划的信贷策略模型得出该银行在年度信贷总额为1亿元时的信贷调整策略。

三、模型假设

关于模型建立上，我们有以下几点假设：

1、信贷风险指数的评定只与信誉评级、是否违约、销项发票有效发票占比、销项发票负数发票占比、企业月均利润有关；

2、信贷策略取决于最大年化利息；

3、贷款额度只与信誉评级与信贷风险指数有关。

四、符号说明

符号	定义
CR_x	第x家企业的信誉等级
BC_x	第x家企业的违约情况
PIR_x	第x家企业的销项发票有效发票占比
PNR_x	第x家企业的销项发票非负数发票占比
AMP_x	第x家企业的月均利润
AMM_x	第x家企业的月利润的最大值（旺季）
AM_x	第x家企业的信贷风险指数

五、模型的建立与求解

5.1 问题一的求解

5.1.1 情景复述

123 家企业的实力、信誉、信贷风险都不一样，银行需依据这些因素来确定是否放贷及贷款额度、利率和期限等信贷策略。因此，需要有一个客观准确的信贷风险指数。而指数的确定需要公平公正，充分考虑每家企业的现实条件与每个指标的重要程度。由此本团队需建立一个模型，以 123 家企业的企业信息、进项发票信息、销项发票信息为数据，去计算每家企业的信贷风险指数，使得信贷风险指数能够合理、公平地反应每个企业的信贷现状。

5.1.2 模型 I：基于主成分分析法的评价指标选择模型

主成分分析也称主分量分析，它通过对原来相关的各原始变量作数学变换，使之成为相互独立的分量，根据每个分量的贡献率选择主分量，然后再对主分量计算综合评价值。使用这种方法的优点在于：消除了评价指标间的相关影响；减少了指标选择的工作量；采用信息量权数，有助于客观地反映样本间的现实关系[3]。

附件 1 内含 3 张表格，第一张表格为企业信息，本文将信誉评级的离散型数据转换为连续型数据，即 A、B、C、D 各自对应 1、0.75、0.5、0.25。将是否违约的判断型数据转换为数字，即数字 0 代表“是”，数字 1 代表“否”。

第二张表格为进项发票信息，分别有企业代号、发票号码、开票日期、销方单位代号、金额、税额、价税合计、发票状态等信息，第三张表格为销项发票信息，从这些信息中，可计算得到每家企业的销项发票有效发票占比以及销项发票非负数发票占比，其公式为：

$$\mathrm{PIR_x} = \frac{\mathrm{EN_x}}{\mathrm{AN_x}}$$

$$\mathrm{PNR_x} = 1 - \frac{\mathrm{NN_x}}{\mathrm{AN_x}}$$

其中，$\mathrm{EN_x}$为每家企业的有效发票数量，$\mathrm{NN_x}$为每家企业的负数发票数量，$\mathrm{AN_x}$为每家企业的总发票数量。

同时，也可将进项发票与销项发票联系起来，从而计算得到每家企业的月均利润和

月利润的最大值（旺季），其公式为：

$$AMP_x = \frac{\sum_i^{n_x}(\sum_j^{m_{xi}} AOM_{xij} - \sum_k^{p_{xi}} TTI_{xij})}{n_x}$$

$$AMM_x = \max(\sum_j^{m_{xi}} AOM_{xij} - \sum_k^{p_{xi}} TTI_{xij})$$

其中，n_x为第 x 家企业的发票所跨越的月份数，m_{xi}为第 x 家企业在第 i 月的销项发票数量，p_{xi}为第 x 家企业在第 i 月的进项发票数量, AOM_{xij}为第 x 家企业在第 i 月的第 j 张发票的销项发票的金额，TTI_{xij}为第 x 家企业在第 i 月的第 j 张发票的进项发票的价税合计。

在得出每家企业的违约情况、销项发票有效发票占比、销项发票非负数发票占比、月均利润、月利润的最大值（旺季）这五个数值后，我们将其作为候选评价指标。

为了选出企业信誉评估的指标，我们利用了主成分分析法，对这五个候选评价指标进行了信息贡献率的计算，从而选择出合理的评价指标。

1. 模型的建立

（1） 标准化处理

假设评价指标变量共有 m 个，记作$x_1, x_2, \dots, x_m$，并且有 n 支待评价的股票，第 i 支股票的第 j 个指标的取值为x_{ij}。将各个指标值x_{ij}转换成标准化指标$\tilde{x}_{ij}$，转换表达式如：

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{S_j}, (i = 1,2, \dots, n; j = 1,2, \dots, m)$$

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$$

$$S_j = \frac{1}{n-1}\sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2, (j = 1, 2, \dots, m)$$

其中，$\bar{x}_j$，S_j为第j个指标的样本均值和标准差。

（2） 相关系数矩阵R的计算

相关系数矩阵$R = (r_{ij})_{m\times m}$，而r_{ij}的计算方式如下所示：

$$r_{ij} = \frac{\sum_{k=1}^{n} \tilde{x}_{ki} \cdot \tilde{x}_{kj}}{n-1}, (i, j = 1, 2, \dots, m)$$

其中$r_{ii} = 1, r_{ij} = r_{ji}$, r_{ij}是第i个指标与第j个指标的相关系数。

（3） 计算特征值和特征向量

接下来计算相关系数矩阵 R 的特征值$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$，以及对应的特征向量$u_1 u_2, \dots, u_m$，其中$u_j = (u_{1j}, u_{2j}, \dots, u_{nj})^T$，由特征向量组成 m 个新的指标变量为：

$$\begin{cases} y_1 = u_{11}\tilde{x}_1 + u_{21}\tilde{x}_2 + \cdots + u_{n1}\tilde{x}_n \\ y_2 = u_{12}\tilde{x}_1 + u_{22}\tilde{x}_2 + \cdots + u_{n2}\tilde{x}_n \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ y_m = u_{1m}\tilde{x}_1 + u_{2m}\tilde{x}_2 + \cdots + u_{nm}\tilde{x}_n \end{cases}$$

其中在表达式中y_1是第1主成分，y_2是第2主成分，依次类推，y_m是第m个主成分。

（4） 计算信息贡献率和累积贡献率

首先需要计算特征值$\lambda_j(j = 1,2,\ldots,m)$的信息贡献率和累积贡献率。而主成分$y_j$的信息贡献率$b_j$计算公式如下：

$$b_j = \frac{\lambda_j}{\sum_{k=1}^{m}\lambda_k}(j = 1,2,\ldots,m)$$

其中b_j为第j个主成分的信息贡献率。

而主成分$y_1, y_2, \ldots, y_p$的累积贡献率计算如下：

$$\alpha_p = \frac{\sum_{k=1}^{p}\lambda_k}{\sum_{k=1}^{m}\lambda_k}$$

当α_p接近于1时，则我们将选择前p个指标变量$y_1, y_2, \ldots, y_p$作为p个主成分，代替原来的m个评价指标变量。

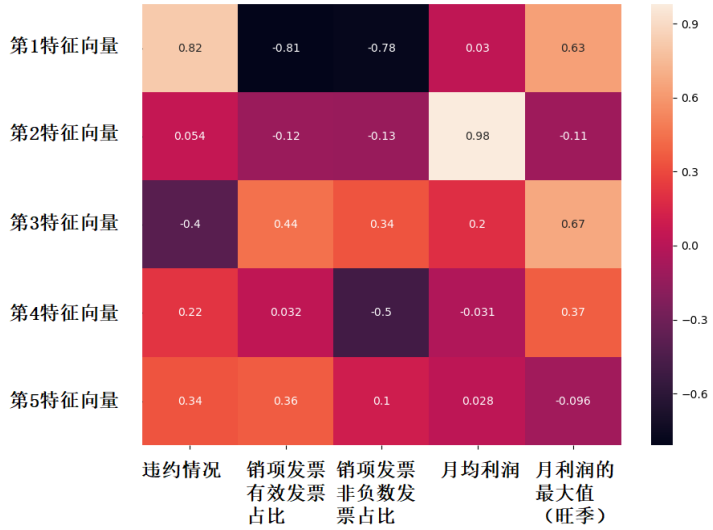
2. 模型的求解

首先对数据进行标准化处理，然后计算每个评价指标之间的相关系数，构成相关系数矩阵，利用相关系数矩阵可计算得出如表1所示的特征向量。

表 1：主成分对应的特征向量

	违约情况	销项发票有效发票占比	销项发票非负数发票占比	月均利润	月利润的最大值（旺季）
第 1 特征向量	0.819	-0.810	0.783	0.03	0.63
第 2 特征向量	0.054	-0.119	-0.129	0.979	-0.11
第 3 特征向量	-0.404	0.443	0.338	0.195	0.665
第 4 特征向量	0.220	0.032	-0.496	-0.031	0.373
第 5 特征向量	0.338	0.365	0.1	0.028	-0.096

图 2：主成分对应的特征向量热力图

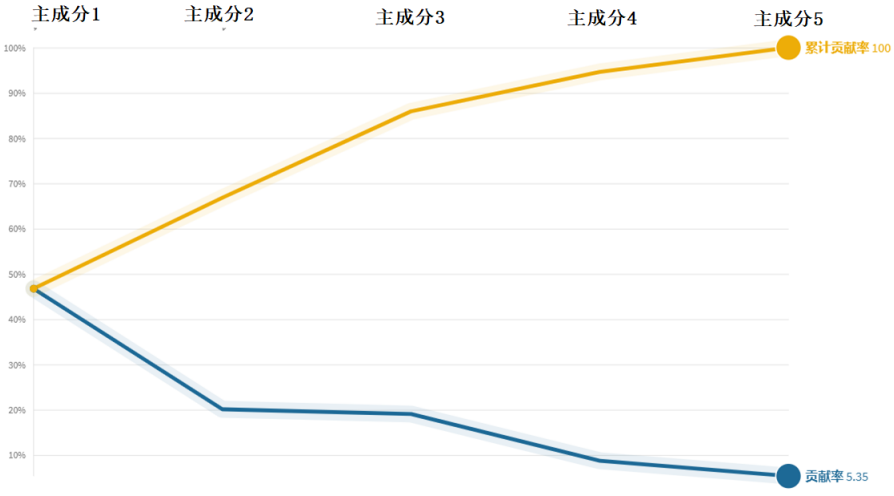


利用特征向量和已标准化后的数据，可以得出对应的主成分y_i，并且利用相关系数矩阵的特征值可求出每个主成分的贡献率以及累计贡献率，结果如表3所示。

表 3：主成分分析结果

主成分序号	贡献率%	累计贡献率%
1	46.758	46.758
2	20.099	66.856
3	19.071	85.927
4	8.721	94.648
5	5.352	100

图 4：主成分分析结果



根据表3的主成分分析结果，前四个主成分累计贡献率超过了94%，因此，我们选取了前四个主成分作为企业信誉评估的指标。

5.1.3 模型 II：基于熵权法和支持向量机的信贷风险指数双重评价模型

熵最先由申农引入信息论，目前已经在工程技术、社会经济等领域得到了非常广泛的应用。熵权法的基本思路是根据指标变异性的大小来确定客观权重。

熵权法以信息熵评价某项指标在不同评价对象中的差异程度，若某项指标的数据变幅很大，说明该指标提供的信息量大，相应的信息熵小，赋予的权重大；反之亦然。若某项指标的数据全部相等，说明该指标对评价对象无法提供区别信息，相应的信息熵最大，权重为零。因此，利用熵权法可客观、合理地得到每个指标的权重，进而得到信贷风险指数评价模型[1]。

SVM 全称为支持向量机（Support Vector Machine）。SVM 非线性回归的基本思想是利用非线性函数φ，将输入变量 x 映射到高维特征空间中[2]，该函数表达式如下:

$$\begin{cases} f(x)=\omega\times\varphi(x)+b,\ f:R^n\to H,\ \omega\in R^n \\ R_s[f]=R_{emp}[f]+\dfrac{1}{2}\|\omega\|^2 \\ g\left(x_i,y_i,f(x_i)\right)=\begin{cases}|f(x)-y|-\varepsilon,\ |f(x)-y|\geq\varepsilon\\ 0,\ |f(x)-y|<\varepsilon\end{cases} \\ f(x)=\omega\times\varphi(x)+b,\ f:R^n\to H,\ \omega\in R^n \end{cases}$$

1. 模型的建立

（1） 数据标准化

将五大指标的数据进行标准化处理。假设给定了q个指标$X_1,X_2,\dots,X_k$，其中：

$$X_i=\{x_1,x_2,\dots,x_n\}$$

假设对各指标数据标准化后的值为$Y_1,Y_2,\dots,Y_k$，那么$Y_{ij}=\frac{X_{ij}-\min(X_i)}{\max(X_i)-\min(X_i)}$，其中，i为第i家企业，j为第j个指标。

（2） 求各指标的信息熵

根据信息论中信息熵的定义，一组数据的信息熵为：

$$E_j=-\frac{1}{\ln n}\sum_{i=1}^{n}p_{ij}\ln p_{ij}$$

其中，$p_{ij}=\frac{Y_{ij}}{\sum_{i=1}^{n}Y_{ij}}$，如果$p_{ij}=0$，则定义$\lim_{p_{ij}\to 0}p_{ij}\ln p_{ij}=0$

（3） 确定各指标权重

根据信息熵的计算公式，计算出各个指标的信息熵为$E_1,E_2,\dots,E_k$。通过信息熵计算各指标的权重：

$$W_i=\frac{1-E_i}{k-\sum E_i}(i=1,2,\dots,k)$$

（4） 企业信贷风险指数的计算

得到各指标的权重后，即可算出企业信贷风险指数：

$$\mathrm{AM_x} = \mathrm{W_1}\overline{\mathrm{BC_x}} + \mathrm{W_2}\overline{\mathrm{PIR_x}} + \mathrm{W_3}\overline{\mathrm{PNR_x}} + \mathrm{W_4}\overline{\mathrm{AMP_x}} + \mathrm{CR_x}$$

其中，$\overline{\mathrm{BC_x}}$、$\overline{\mathrm{PIR_x}}$、$\overline{\mathrm{PNR_x}}$、$\overline{\mathrm{AMP_x}}$为标准化后的数据。

（5） SVM函数线性回归

我们对上述函数进行线性回归，权值向量是ω，偏差值是b，并且都可以通过最小化结构风险求解得到，表达式如下：

$$R_s[f] = R_{emp}[f] + \frac{1}{2}\|\omega\|^2 = \frac{1}{I}\sum_{i=1}^{l} g(x_i, y_i, f(x_i)) + \frac{1}{2}\|\omega\|^2$$

其中，$\mathrm{R_{emp}[f]}$是经验风险，$\frac{1}{2}\|\omega\|^2$是置信区间。结构风险是期望风险的一个上界。$\mathrm{g(x_i, y_i, f(x_i))}$是损失函数，给定损失函数，则结构风险最小化问题可以作为一个二次规划问题求解。而 Vapnik 提出ε不敏感损失函数，该函数具有较好的稀疏性，可以保证结果的泛化性。损失函数如下：

$$g\big(x_i, y_i, f(x_i)\big) = \begin{cases} |f(x) - y| - \varepsilon, & |f(x) - y| \geq \varepsilon \\ 0, & |f(x) - y| < \varepsilon \end{cases}$$

其中，ε是大于零的参数，控制支持向量的个数和泛化能力。ε的值越小，精度越高，支持向量数越多，但是泛化能力越弱。而最小化问题可以转化为如下表达式：

$$\begin{cases} min\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{l}(\xi + \xi^*) \\ s.t.\, y_i - \omega \times (x_i) - b \leq \varepsilon + \xi_i \\ \omega \times (x_i) + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0,\ \ i = 1, \ldots, l \end{cases}$$

其中，引入非负松弛变量ξ和ξ*，用于度量ε⁻不敏感带外的训练样本的偏离程度，并引入一个大于零的惩罚参数 C，用于控制拟合程度。并将原问题转化为对偶问题，改写成凸二次规划如下所示：

$$\min\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i^* - \alpha_i)\big(\alpha_j^* - \alpha_j\big)K(x_i, x_j)$$

$$+\varepsilon\sum_{i=1}^{l}(\alpha_i^* - \alpha_i) - \sum_{i=1}^{l} y_i(\alpha_i^* - \alpha_i)$$

$$\mathrm{s.t.}\sum_{i=1}^{l}(\alpha_i^* - \alpha_i) = 0$$

$$0 \leq \alpha_i^*, \alpha_i \leq C, i = 1, \ldots, l$$

其中，$K\big(x_i, x_j\big) = \varphi(x_i) \times \varphi(x_j)$为核函数，公式如下所示：

$$K(x_i, x_j) = \exp(-\|x_i - x_j\|^2/\sigma^2)$$

σ是该核函数的参数，因此可求得SVM预测模型为：

$$f(x) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)K(x_i, x) + b$$

2. 模型的求解

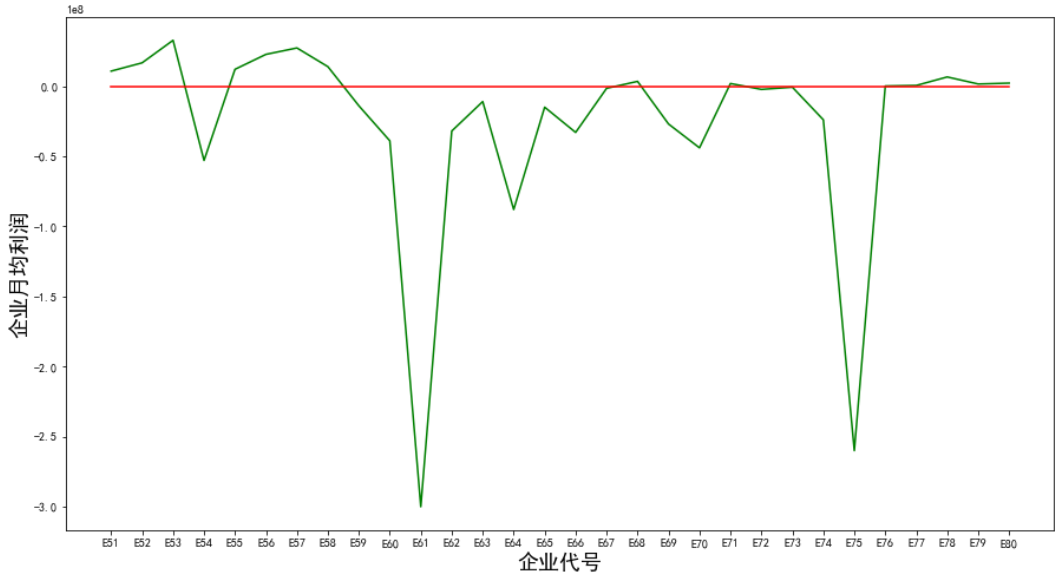
根据所建立的基于熵权法的信贷风险指数评价模型，将各个指标进行数据标准化，得到的数据如下表所示：

表 5：部分企业的各个指标数据标准化后的表

企业代号	违约情况	销项发票有效发票占比	销项发票非负数发票占比	月均利润
E1	1.0	0.9600	0.04094	0.0
E2	1.0	0.8811	0.0418	0.7173
E3	1.0	0.9768	0.2711	0.5495
E4	1.0	0.8765	0.0058	1.0
E5	1.0	0.9248	0.0137	0.6257
E6	1.0	0.8084	0.0317	0.6693
…	…	…	…	…
E118	0.0	0.9194	0.0101	0.5893
E119	0.0	0.7929	0.1381	0.5894
E120	0.0	0.0	1.0	0.5895
E121	0.0	0.8207	0.1169	0.5889
E122	0.0	0.8034	0.0983	0.5879
E123	0.0	0.2862	0.5800	0.5890

为了解决各指标不同量纲无法进行直接汇总的问题,对处理后的数据进行去量纲操作——标准化，使得数据更加平稳，而不会有太大的波动，这样便于计算信息熵。以下为“企业月均利润”指标标准化前和标准化后的对比图，可以看出数据在标准化前和标准化后的数据有着明显的差别。

图 6：“企业月均利润”指标标准化前后的对比图



根据信息熵的计算公式，可以计算出4项指标各自的信息熵如下：

表 7：4 项指标信息熵表

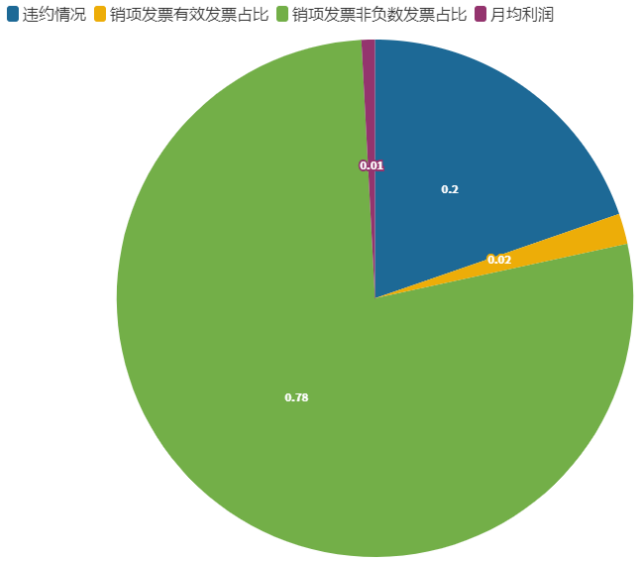
	违约情况	销项发票有效发票占比	销项发票非负数发票占比	月均利润
信息熵	0.9485	0.9950	0.7974	0.9978

得到各项指标的信息熵后，便可根据指标权重的计算公式，可以得到各个指标权重如下表所示：

表 8：4 项指标的权重表

	W_1	W_2	W_3	W_4
权重	0.1971	0.0191	0.7754	0.0084

图 9：4 项指标的权重饼图



最后，便可计算出每家企业的信贷风险指数，如下表所示：

表 10：部分企业的信贷风险指数表

企业代号	信贷风险指数	企业代号	信贷风险指数	企业代号	信贷风险指数
E1	1.247206	E8	1.262255	E15	1.219817
E2	1.252372	E9	1.244181	E16	1.246671
E3	0.763949	E10	0.893877	E17	1.24007
E4	0.560152	E11	0.594409	E18	1.224828
E5	0.897328	E12	0.890085	E19	1.242963
E6	1.242775	E13	1.321851	E20	0.888233
E7	1.485176	E14	0.567397	E21	0.92262

训练 SVM 预测模型的关键参数如下，核函数类型选择 SVC，将样本数据划分为 83%为训练集，17%为测试集。比计算均方误差（MSE）、平均绝对误差（MAE）进行模型评价，得到的模型评价结果如下：

表 11：MSE 与 MAE 评价的模型评价结果

模型	MSE	MAE
SVM	1.3157	3.2608

5.1.4 模型 III：基于线性规划的信贷策略模型

1. 模型的建立

为了确定是否给企业放贷，本文设置了一个信贷风险指数阈值，该阈值取值大小为所有企业的信贷风险指数的平均数，即：

$$\lambda = \frac{\sum_x^n AM_x}{n}$$

其中，n为企业数量。

确定给能够放贷的企业后，接下来需要得到贷款额度，其额度的计算规则为：

对于信誉评级为A的企业的贷款额度：$\mathrm{LL_x} = \alpha \times \frac{3\times \mathrm{AN}}{3\times \mathrm{AN}+2\times \mathrm{BN}+\mathrm{CN}} \times \frac{\mathrm{AM_x}}{\sum_{\mathrm{i}}^{\mathrm{AN}} \mathrm{AM_i}}$

对于信誉评级为B的企业的贷款额度：$\mathrm{LL_x} = \alpha \times \frac{2\times \mathrm{BN}}{3\times \mathrm{AN}+2\times \mathrm{BN}+\mathrm{CN}} \times \frac{\mathrm{AM_x}}{\sum_{\mathrm{i}}^{\mathrm{BN}} \mathrm{AM_i}}$

对于信誉评级为C的企业的贷款额度：$\mathrm{LL_x} = \alpha \times \frac{\mathrm{CN}}{3\times \mathrm{AN}+2\times \mathrm{BN}+\mathrm{CN}} \times \frac{\mathrm{AM_x}}{\sum_{\mathrm{i}}^{\mathrm{CN}} \mathrm{AM_i}}$

其中，α为银行的信贷总额，AN为可放贷的信誉评级为A的企业的数量，BN为信誉评级为B的企业，CN为信誉评级为C的企业。

为得到信贷策略，建立线性规划模型：

$$\mathrm{MAI} = (\left(\sum_i^{AN} LL_i\right) \times (1 - WR_1) + \left(\sum_i^{BN} LL_i\right) \times (1 - WR_2)$$
$$+ \left(\sum_i^{CN} LL_i\right) \times (1 - WR_3)) \times \mathrm{IR}$$

其中，MAI为最大年化利息，$\mathrm{LL_i}$为第 i 家企业的贷款额度，IR为利率，$\mathrm{WR_1}$、$\mathrm{WR_2}$、$\mathrm{WR_3}$为信誉等级 A、B、C 的企业流失率。利率和企业流失率的约束条件如附件 3 中银行贷款年利率与客户流失率关系的 2019 年统计数据所示。

2.模型的求解

对于附件1中的123家企业，银行的信贷风险指数阈值为0.745，即能够贷款的企业数量为69家，其中，这些企业的贷款额度如下表所示：

表 12：部分企业的贷款额度表

企业代号	贷款额度	企业代号	贷款额度	企业代号	贷款额度
E1	0.0157 α	E9	0.0157 α	E18	0.0154 α
E2	0.0158 α	E10	0.0104 α	E19	0.0157 α
E3	0.0066 α	E12	0.0103 α	E20	0.0103 α
E5	0.0104 α	E13	0.0166 α	E21	0.0107 α
E6	0.0157 α	E15	0.0154 α	E22	0.0154 α
E7	0.0187 α	E16	0.0157 α	E23	0.0106 α
E8	0.0159 α	E17	0.0156 α	E24	0.0156 α

经计算可得到，对于附件 1 中 123 家企业，当利率为 0.04 时，信誉评分为 A、B、C 的客户流失率都为 0，此时年化利息最大，即此时的信贷策略是最优的。

5.2 问题二的求解

5.2.1 模型的建立

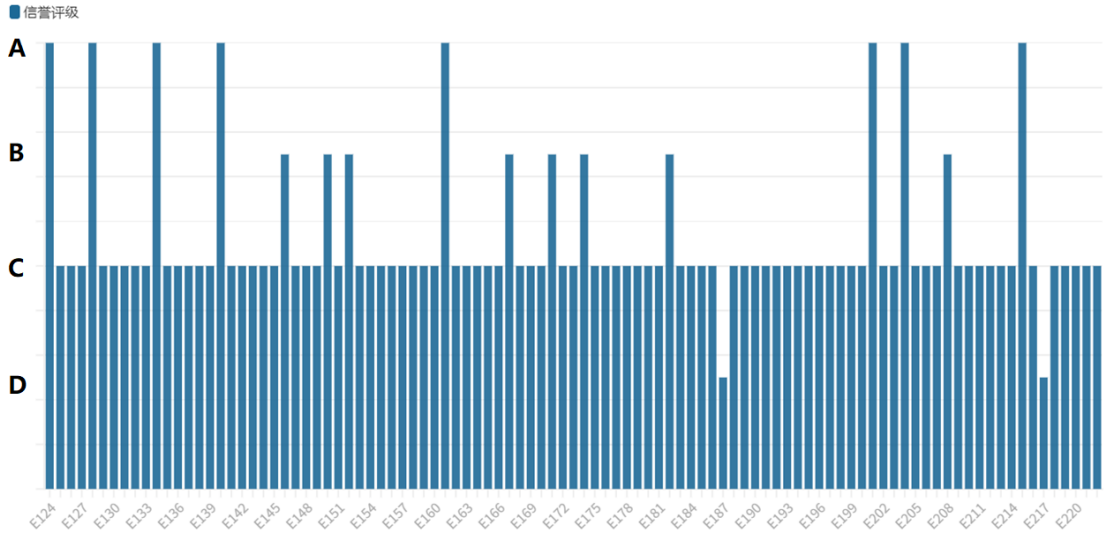
本问的信贷策略模型基于问题一所构建的基于线性规划的信贷策略模型，通过问题一的信贷风险指数评价模型计算出每家企业的信贷风险指数，利用信贷风险指数阈值 $\lambda = \frac{\sum_{x}^{n} AM_x}{n}$得出为 1.47，即能够贷款的企业数量为 143 家。

由问题一的SVM模型，代入附件二，得出各企业风险等级如下：

表13：附件2部分企业风险等级

企业代号	风险等级	企业代号	风险等级	企业代号	风险等级
E124	A	E136	C	E147	C
E127	C	E138	C	E148	C
E128	A	E139	C	E150	B
E131	C	E140	A	E151	C
E132	C	E142	C	E152	B
E134	A	E146	B	E153	C

图14：附件2前一百个企业风险等级



信誉等级A、B、C的企业放款额度，由题意可知对信誉评级为D的企业原则上不予放贷。我们建立以下关于银行年化利息的线性规划方程:

$$LL_A = \alpha \times \frac{3 \times AN}{3 \times AN + 2 \times BN + CN} \times \frac{AM_x}{\sum_i^{AN} AM_i}$$

$$LL_B = \alpha \times \frac{2 \times BN}{3 \times AN + 2 \times BN + CN} \times \frac{AM_x}{\sum_i^{BN} AM_i}$$

$$LL_C = \alpha \times \frac{CN}{3 \times AN + 2 \times BN + CN} \times \frac{AM_x}{\sum_i^{CN} AM_i}$$

$$\mathrm{MAI} = (\left(\sum_i^{AN} LL_i\right) \times (1 - WR_1) + \left(\sum_i^{BN} LL_i\right) \times (1 - WR_2) + \left(\sum_i^{CN} LL_i\right) \times (1 - WR_3)) \times \mathrm{IR}$$

由查阅资料，我们不妨设其中 A、B、C 等级的权重为 3:2:1，同时同一等级的额度由不同的风险指数确定。由题意可知，银行年度信贷总额为 1 亿元，对确定要放贷企业的贷款额度为10~100万元，所以信贷上限为 100 万元，下限为 10 万元。当理想放贷额大于 100 万元时，则只借贷 100 万元，超出部分借贷给其他企业；当理想放贷额小于 10 万元时，则补齐至 10 万元。

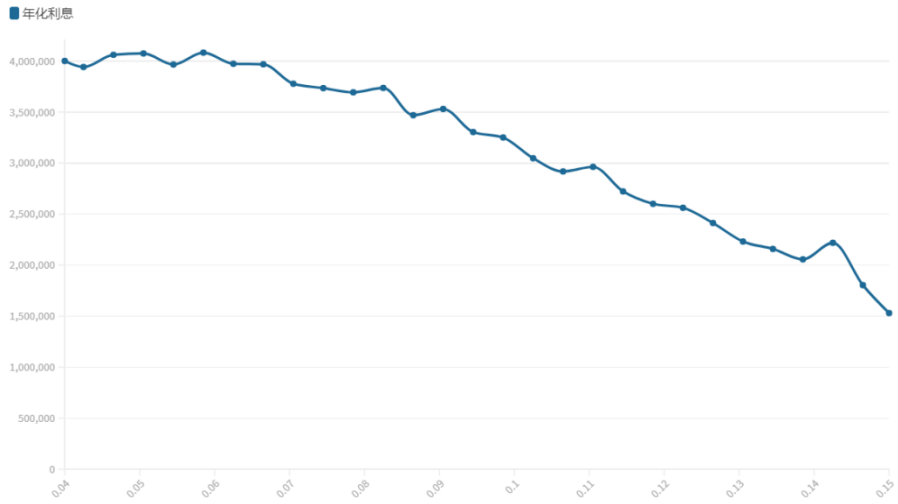
这些企业的贷款额度如下表所示：

表 15：附件 2 部分可贷款企业的贷款额度表（单位：万元）

企业代号	贷款额度	企业代号	贷款额度	企业代号	贷款额度
E124	100	E136	51.08	E147	50.78
E127	51.23	E138	50.84	E148	50.9
E128	100	E139	51.04	E150	100
E131	50.59	E140	100	E151	51.23
E132	52.01	E142	50.83	E152	100
E134	100	E146	100	E153	60.6

由银行年化利息的线性规划方程经计算可得到，对于附件2中可借贷的143家企业，当**利率为0.0585**时，信誉评分为A、B、C的客户流失率分别为0.347、0.302、0.29，**此时年化利息为408.16万元，即此时的信贷策略是最优的**。

图 16：银行在年度信贷总额为 1 亿元时年化利息曲线



5.2 问题三的求解

企业的生产经营和经济效益可能会受到一些突发因素影响，而且突发因素往往对不同行业、不同类别的企业会有不同的影响，特别是 2020 年初新冠病毒疫情的爆发，许多企业的经营受到严重影响。

表 17：GDP 同比增长速度　　单位：%

年份	1 季度	2 季度	3 季度	4 季度
2015	7.1	7.1	7.0	6.9
2016	6.9	6.8	6.8	6.9
2017	7.0	7.0	6.9	6.8
2018	6.9	6.9	6.7	6.5
2019	6.4	6.2	6.0	6.0
2020	-6.8			

注：同比增长速度为与上年同期对比的增长速度

表 18：GDP 环比增长速度　　单位：%

年份	1 季度	2 季度	3 季度	4 季度
2015	1.8	1.8	1.7	1.6
2016	1.5	1.8	1.7	1.6
2017	1.7	1.8	1.6	1.5
2018	1.7	1.7	1.5	1.5
2019	1.6	1.5	1.3	1.5
2020	-6.8			

注：同比增长速度为与上年同期对比的增长速度

由上表可知，因疫情影响，GDP增长速度受到了影响，这充分说明企业的运营受到了疫情的冲击。

表19：2020一季度GDP初步核算数据

	绝对额（亿元）	比上年同期增长（%）
	1 季度	1 季度
GDP	206504	-6.8
第一产业	10186	-3.2
第二产业	73638	-9.6
第三产业	122680	-5.2
农林牧渔业	10708	-2.8
工业	64642	-8.5
制造业	53852	-10.2
批发和零售业	18750	-17.8
交通运输、仓储和邮政业	7865	-14.0
住宿和餐饮业	2821	-35.3
金融业	21347	6.0
信息传输、软件和信息技术	8928	13.2

由上图可知，不同类型的企业受疫情这种突发情况的影响程度也不同。其中，个体经济受影响最大，此时该行业抵抗金融冲击能力也降低，无形中提高了客户流失率，也打击了贷款偿还能力。相对的，银行也会适当降低该信誉等级，缩减可信贷金额。

利用问题一的模型，我们不妨令个体经营户增加30%的销项发票、发票量减少40%，即减少了交易量与营收额，同时适当提高各利率的客户流失率，对附件2进行处理后得出下表：

表20：附件2部分可贷款企业的受影响贷款额度表（单位：万元）

企业代号	贷款额度	企业代号	贷款额度	企业代号	贷款额度
*E124	97.3	E136	51.08	E147	51.78
*E127	34.23	*E138	34.84	E148	48.9
*E128	92.32	*E139	33.04	E150	100
*E131	33.59	E140	100	E151	50.23
*E132	35.01	#E142	67.83	#E152	100
E134	100	E146	100	*E153	37.6

新冠疫情暴发后，国家及时启动一级应急响应方案，采取了疫情防控、救助救援、民生保障、社会安稳、经济刺激等一系列措施，对有效防控疫情起到了积极作用。但同时新冠疫情对企业也带来了巨大的风险隐患和不确定性[4]。

其中，带星号的均为易受疫情影响的个体经营企业，经计算约下降了30%的贷款额度，带井号的为疫情影响有促进效果的企业，约有15%的信用增幅。

同理，各自突发情况对不同的企业有不同的影响，仅看2020年新冠疫情，绝大部分产业负增长，仅有少部分企业例如信息、金融、物流等仍保持小幅度增长。负增长行业营收额降低、销项发票数量变大等，使得信贷风险指标降低，进而导致信贷额度减少。

利用问题一所构建的基于线性规划的信贷策略模型，与问题二同理，由表16中的新信贷额度，经过银行年化利息的线性规划方程经计算可得到，对于附件2中可借贷的143家企业，当**利率为0.04**时，信誉评分为A、B、C的客户流失率均为0，**此时年化利息为400万元，即此时的信贷策略是最优的。**

六、模型的评价与推广

6.1 模型的优点

(1) **针对模型Ⅰ**，基于主成分分析法的评价指标选择模型客观合理简便地分析各个候选评价指标的适用性，并根据各个指标的信息贡献率从五个候选指标中选出四个最为重要的评价指标。
(2) **针对模型Ⅱ**，基于熵权法和支持向量机的信贷风险指数双重评价模型以数据来说明每个指标的重要程度，而不以主观意识为主导。熵权法和支持向量机结合让信贷风险指数模型的计算结果更加令人信服，所计算得到的指数也更能够反映企业信贷的真实情况。
(3) **针对模型Ⅲ**，基于线性规划的信贷策略模型充分考虑每家企业的实力、信誉，以年利息最大化为银行的首要目的，模型符合真实场景，能够较为准确、合理地得出信贷策略。

6.2 模型的缺点

(1) **针对模型Ⅰ**，对于在非高斯分布情况下，主成分分析方法得出的主元可能并不是最优的。而且如果用户对观测对象有一定的先验知识，掌握了数据的一些特征，却无法通过参数化等方法对处理过程进行干预,可能会得不到预期的效果,效率也不高。
(2) **针对模型Ⅱ**，熵权法对样本的依赖性较大，随着建模样本不断变化，权重会发生一定波动。因此，权重可能会有一定的偏差。
(3) **针对模型Ⅲ**，影响信贷策略的复杂因素较多，而基于线性规划的信贷策略模型不能对其进行全面的考虑，造成与实际有一定的不相符。

6.3 模型的改进

(1) **模型Ⅱ**考虑了四个评价指标构成的信贷风险评价指标体系，来评价银行对于每家企业的信贷风险。这主要是依据附件 1 来考虑的，可以尝试采用更多更有效的指标来评价模型，从而让模型达到更加优化的目的。
(2) **模型III**只以最大年化利息为信贷策略是否优劣的评判标准，对于影响因素甚多的信贷策略模型，可能得到的信贷策略并不是最优的，可以考虑更多的因素来建立信贷策略模型，进而让模型结果更具可靠性。

6.4 模型的推广

(1) 本文构建了基于熵权法和支持向量机的信贷风险指数双重评价模型，解决了企业信贷风险评价的问题，采用标准化的数据表示相关指标，具有一定的合理性，解决了各指标不同量纲无法进行直接汇总的问题。
(2) 本文提出的基于线性规划的信贷策略模型具有良好的应用前景，可以和基于熵权法和支持向量机的信贷风险指数双重评价模型相结合，得出更加可靠的结果。

七、参考文献

[1]仇军,卓飞,李岩.基于 AHP—熵权法的城乡供水一体化评估模型[J].广东水利水电,2020(07):16-21.
[2]郭建利，程蕾，孙博超，颜瑞。基于 ARIMA-SVM 的煤炭价格预测及实证研究，煤炭经济研究，第 36 卷第 2 期：6-10
[3]张子夜. 基于主成分分析法对信贷风险的综合评估[J]. 金融理论探索,2011(3):43-44.
[4]刘淑玲.新冠疫情下企业财务预警模式研究[J].企业改革与管理,2020,(12):143-144.
[5]http://www.gov.cn/xinwen/2020-04/18/content_5503803.htm

八、附录

附录 1：Python：task1.py

```
# -*- coding: gb2312 -*-
import pandas as pd
import xlsxwriter
import math

def defense_condition():
    workbook = xlsxwriter.Workbook('NO_1.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "是否违约")

    excel = pd.read_excel('附件 1：123 家有信贷记录企业的相关数据.xlsx')
    datas = excel['是否违约']
    line = 1
    for data in datas:
        if (data == '是'):
            worksheet.write(line, 0, "1")
        else:
            worksheet.write(line, 0, "0")
        line += 1
    workbook.close()

def credit():
    workbook = xlsxwriter.Workbook('NO_2.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "信誉评级")

    excel = pd.read_excel('附件 1：123 家有信贷记录企业的相关数据.xlsx')
    datas = excel['信誉评级']
    line = 1
    for data in datas:
        if (data == 'A'):
            worksheet.write(line, 0, "1")
        elif(data=='B'):
            worksheet.write(line, 0, "0.75")
        elif(data=='C'):
            worksheet.write(line,0,"0.5")
        elif(data=='D'):
            worksheet.write(line,0,"0.25")
        line += 1
    workbook.close()

def is_valid():
    workbook = xlsxwriter.Workbook('NO_2.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "销项发票状态")

    excel = pd.read_excel('附件 1：123 家有信贷记录企业的相关数据.xlsx')
    datas = excel['发票状态']
    line = 1
    for data in datas:
        if (data == '有效发票'):
            worksheet.write(line, 0, "1")
        else:
            worksheet.write(line, 0, "0")
```

```
            line += 1
        workbook.close()

def valid_rate():
    excel_1 = pd.read_excel('企业信息.xlsx')
    excel_2 = pd.read_excel('进项发票信息.xlsx')
    excel_3=pd.read_excel('销项发票信息.xlsx')
    workbook = xlsxwriter.Workbook('NO_3.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "销项 rate")

    sale_id=excel_3['企业代号']
    sale_state=excel_3['发票状态']
    now_id=sale_id[0]
    valid=0
    invalid=0
    line=1
    for i in range(len(sale_id)):
        if  sale_id[i]==now_id:
            if sale_state[i]=='有效发票':
                valid+=1
            else:
                invalid+=1
        else:
            if(valid+invalid==0):
                print(0)
            else:
                worksheet.write(line,0,valid/(valid+invalid))
                line+=1
                print(valid/(valid+invalid))
            valid=0
            invalid=0
            if sale_state[i]=='有效发票':
                valid=1
            else:
                invalid=1
            now_id = sale_id[i]

        if i == len(sale_id) - 1:
            print(valid/(valid+invalid))
            worksheet.write(line,0,valid/(valid+invalid))
    workbook.close()

def negative_rate():
    excel_1 = pd.read_excel('企业信息.xlsx')
    excel_2 = pd.read_excel('进项发票信息.xlsx')
    excel_3=pd.read_excel('销项发票信息.xlsx')
    workbook = xlsxwriter.Workbook('NO_4.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "销项 rate")

    sale_id=excel_3['企业代号']
    sale_state=excel_3['金额']
    now_id=sale_id[0]
    valid=0
    invalid=0
    line=1
    for i in range(len(sale_id)):
```

```
        if  sale_id[i]==now_id:
            if sale_state[i]>0:
                valid+=1
            else:
                invalid+=1
        else:
            if(valid+invalid==0):
                print(0)
            else:
                worksheet.write(line,0,valid/(valid+invalid))
                line+=1
                print(valid/(valid+invalid))
            valid=0
            invalid=0
            if sale_state[i]>0:
                valid=1
            else:
                invalid=1
            now_id=sale_id[i]

        if i == len(sale_id) - 1:
            print(valid/(valid+invalid))
            worksheet.write(line,0,valid/(valid+invalid))
    workbook.close()

def per_month_profit():
    excel_1 = pd.read_excel('企业信息.xlsx')
    excel_2 = pd.read_excel('进项发票信息.xlsx')
    excel_3=pd.read_excel('销项发票信息.xlsx')
    workbook = xlsxwriter.Workbook('NO_5.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "销项 rate")

    sale_id=excel_3['企业代号']
    sale_state=excel_3['金额']
    price_tex_sum=excel_2['价税合计']

    now_id=sale_id[0]
    total=0
    line=1
    for i in range(len(sale_id)):
        if  sale_id[i]==now_id:
            total+=sale_state[i]-price_tex_sum[i]
        else:
            print(total)
            worksheet.write(line,0,total)
            line+=1
            total=0
        if i==len(sale_id)-1:
            print(total)
            worksheet.write(line,0,total)
        now_id=sale_id[i]
    workbook.close()

def calculate_average():
    excel = pd.read_excel('raw_data.xlsx')
    avg_valid=excel['是否违约']
    avg_credit=excel['信誉评级']
    avg_valid_rate = excel['销项发票有效发票占比']
```

```
    avg_nagetive_rate = excel['销项发票负数发票占比']
    avg_profit=excel['企业月均利润']
    avg_1=0
    avg_2=0
    avg_3=0
    avg_4=0
    avg_5=0
    for i in range(len(avg_valid)):
        avg_1+=avg_valid[i]
        avg_2+=avg_credit[i]
        avg_3+=avg_valid_rate[i]
        avg_4+=avg_nagetive_rate[i]
        avg_5+=avg_profit[i]
    print(avg_1/len(avg_valid))
    print(avg_2 / len(avg_valid))
    print(avg_3 / len(avg_valid))
    print(avg_4 / len(avg_valid))
    print(avg_5 / len(avg_valid))


def calculate_standard():
    excel = pd.read_excel('raw_data.xlsx')
    avg_valid=excel['是否违约']
    avg_credit=excel['信誉评级']
    avg_valid_rate = excel['销项发票有效发票占比']
    avg_nagetive_rate = excel['销项发票负数发票占比']
    avg_profit=excel['企业月均利润']

    avg_1 = excel['平均值_是否违约'][0]
    avg_2 = excel['平均值_信誉评级'][0]
    avg_3 = excel['平均值_销项发票有效发票占比'][0]
    avg_4 = excel['平均值_销项发票负数发票占比'][0]
    avg_5 = excel['平均值_企业月均利润'][0]

    std_1=0
    std_2=0
    std_3 = 0
    std_4 = 0
    std_5 = 0

    for i in range(len(avg_valid)):
        std_1+=(avg_valid[i]-avg_1)*(avg_valid[i]-avg_1)
        std_2 += (avg_credit[i] - avg_2) * (avg_credit[i] - avg_2)
        std_3 += (avg_valid_rate[i] - avg_3) * (avg_valid_rate[i] - avg_3)
        std_4 += (avg_nagetive_rate[i] - avg_4) * (avg_nagetive_rate[i] - avg_4)
        std_5 += (avg_profit[i] - avg_5) * (avg_profit[i] - avg_5)

    print(std_1)
    print(std_2)
    print(std_3)
    print(std_4)
    print(std_5)

def dispose_standard():
    excel = pd.read_excel('raw_data.xlsx')
    valid = excel['是否违约']
    credit = excel['信誉评级']
    valid_rate = excel['销项发票有效发票占比']
```

```
    nagetive_rate = excel['销项发票负数发票占比']
    profit = excel['企业月均利润']

    avg_1 = excel['平均值_是否违约'][0]
    avg_2 = excel['平均值_信誉评级'][0]
    avg_3 = excel['平均值_销项发票有效发票占比'][0]
    avg_4 = excel['平均值_销项发票负数发票占比'][0]
    avg_5 = excel['平均值_企业月均利润'][0]

    std_1 = excel['标准差_是否违约'][0]
    std_2 = excel['标准差_信誉评级'][0]
    std_3 = excel['标准差_销项发票有效发票占比'][0]
    std_4 = excel['标准差_销项发票负数发票占比'][0]
    std_5 = excel['标准差_企业月均利润'][0]

    workbook = xlsxwriter.Workbook('标准化后的数据.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "是否违约")
    worksheet.write(0, 1, "信誉评级")
    worksheet.write(0, 2, "销项发票有效发票占比")
    worksheet.write(0, 3, "销项发票负数发票占比")
    worksheet.write(0, 4, "企业月均利润")

    line=1

    for i in range(len(valid)):
        worksheet.write(line,0,str((valid-avg_1)/std_1))
        worksheet.write(line, 1, str((credit - avg_2) / std_2))
        worksheet.write(line, 2, str((valid_rate - avg_3) / std_3))
        worksheet.write(line, 3, str((nagetive_rate - avg_4) / std_4))
        worksheet.write(line, 4, str((profit - avg_5) / std_5))
    workbook.close()

def standard_weight():
    workbook = xlsxwriter.Workbook('file/weight 标准化后的数据.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "是否违约")
    worksheet.write(0, 1, "信誉评级")
    worksheet.write(0, 2, "销项发票有效发票占比")
    worksheet.write(0, 3, "销项发票非负数发票占比")
    worksheet.write(0, 4, "企业月均利润")

    excel=pd.read_excel('file/change_raw_data.xlsx')
    min_1=excel['min_1'][0]
    min_2 = excel['min_2'][0]
    min_3 = excel['min_3'][0]
    min_4 = excel['min_4'][0]
    min_5 = excel['min_5'][0]

    max_1=excel['max_1'][0]
    max_2 = excel['max_2'][0]
    max_3 = excel['max_3'][0]
    max_4 = excel['max_4'][0]
    max_5 = excel['max_5'][0]

    valid = excel['是否违约']
    credit = excel['信誉评级']
```

```
    valid_rate = excel['销项发票有效发票占比']
    nagetive_rate = excel['销项发票非负数发票占比']
    profit = excel['企业月均利润']


    line=1
    for i in range(len(valid)):
        worksheet.write(line,0,str((valid[i]-min_1)/(max_1-min_1)))
        worksheet.write(line, 1, str((credit[i] - min_2) / (max_2- min_2)))
        worksheet.write(line, 2, str((valid_rate[i] - min_3) / (max_3 - min_3)))
        worksheet.write(line, 3, str((nagetive_rate[i] - min_4) / (max_4 -
min_4)))
        worksheet.write(line, 4, str((profit[i] - min_5) / (max_5 - min_5)))
        line+=1
    workbook.close()

def get_p():
    excel = pd.read_excel('file/weight 标准化后的数据.xlsx')
    valid = excel['是否违约']
    credit = excel['信誉评级']
    valid_rate = excel['销项发票有效发票占比']
    nagetive_rate = excel['销项发票非负数发票占比']
    profit = excel['企业月均利润']

    #先求总和
    all_1=0
    all_2=0
    all_3=0
    all_4=0
    all_5=0
    for i in range(len(valid)):
        all_1+=valid[i]
        all_2+=credit[i]
        all_3+=valid_rate[i]
        all_4+=nagetive_rate[i]
        all_5+=profit[i]

    workbook = xlsxwriter.Workbook('file/p.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "p_是否违约")
    worksheet.write(0, 1, "p_信誉评级")
    worksheet.write(0, 2, "p_销项发票有效发票占比")
    worksheet.write(0, 3, "p_销项发票负数发票占比")
    worksheet.write(0, 4, "p_企业月均利润")

    print(all_1,all_2,all_3,all_4,all_5)

    line=1
    sum_p1=0
    sum_p2=0
    sum_p3=0
    sum_p4=0
    sum_p5=0

    for i in range(len(valid)):
        p1=valid[i]/all_1
        p2=credit[i] / all_2
        p3=valid_rate[i] / all_3
```

```
        p4=nagetive_rate[i] / all_4
        p5=profit[i] / all_5

        if p1!=0:
            sum_p1+=p1*math.log(p1)
        if p2!=0:
            sum_p2 += p2*math.log(p2)
        if p3!=0:
            sum_p3 += p3*math.log(p3)
        if p4!=0:
            sum_p4 += p4*math.log(p4)
        if p5!=0:
            sum_p5 += p5*math.log(p5)


        line+=1
    n=len(valid)
    print(sum_p1, sum_p2, sum_p3, sum_p4, sum_p5)
    print("-----------")

    print(-sum_p1/math.log(n))
    print(-sum_p2 / math.log(n))
    print(-sum_p3 / math.log(n))
    print(-sum_p4 / math.log(n))
    print(-sum_p5 / math.log(n))

def get_weight():
    excel = pd.read_excel('file/weight 标准化后的数据.xlsx')
    valid = excel['是否违约']
    credit = excel['信誉评级']
    valid_rate = excel['销项发票有效发票占比']
    nagetive_rate = excel['销项发票非负数发票占比']
    profit = excel['企业月均利润']

    E1=0.9485
    E2=0.9950
    E3=0.7974
    E4=0.9978

    E=E1+E2+E3+E4
    k=4
    w1=(1-E1)/(k-E)
    w2 = (1 - E2) / (k - E)
    w3 = (1 - E3) / (k - E)
    w4 = (1 - E4) / (k - E)

    print(w1,w2,w3,w4)

    workbook = xlsxwriter.Workbook('file/风险指数.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0,0,"企业代号")
    line =1
    for i in range(len(valid)):

danger=w1*valid[i]+w2*valid_rate[i]+w3*nagetive_rate[i]+w4*profit[i]+credit
[i]
        worksheet.write(line,0,danger)
        line+=1
    workbook.close()
```

```
def change_raw_data():
    excel=pd.read_excel('raw_data.xlsx')
    valid=excel['是否违约']
    valid_rate=excel['销项发票负数发票占比']

    workbook = xlsxwriter.Workbook('file/change_raw_data.xlsx')
    worksheet = workbook.add_worksheet()
    line=1
    for i in range(len(valid)):
        if valid[i]==0:
            v=1
        elif valid[i]==1:
            v=0
        r=1-valid_rate[i]
        worksheet.write(line,3,v)
        worksheet.write(line,5,r)
        line+=1
    workbook.close()

#calculate limit money
def limit_money():
    workbook = xlsxwriter.Workbook('file/limit_money_all.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0,0,'企业代号')
    worksheet.write(0,1,'贷款额度')

    level_a=0
    level_b=0
    level_c=0
    #the number of level A/B/C company
    excel=pd.read_excel("企业信息.xlsx")
    level=excel['信誉评级']
    id=excel['企业代号']

    danger_index = excel['风险指数']
    danger_a=0
    danger_b=0
    danger_c=0

    for i in range(len(level)):
        if level[i]=='A':
            level_a+=1
            danger_a+=danger_index[i]
        elif level[i]=='B':
            level_b+=1
            danger_b += danger_index[i]
        elif level[i]=='C':
            level_c+=1
            danger_c += danger_index[i]
    para_all=3*level_a+2*level_b+level_c
    para_a=(3*level_a)/para_all
    para_b=(2*level_b)/para_all
    para_c=level_c/para_all


    line=1
    for i in range(len(danger_index)):
        #if danger_index[i] > 0.745:
```

```
        worksheet.write(line,0,str(id[i]))
        if level[i]=='A':
            worksheet.write(line,1,str(para_a*danger_index[i]/danger_a))
        if level[i]=='B':
            worksheet.write(line,1,str(para_b*danger_index[i]/danger_b))
        if level[i]=='C':
            worksheet.write(line,1,str(para_c*danger_index[i]/danger_c))
        line+=1

    workbook.close()

def get_MAI():
    excel = pd.read_excel('file/limit_money_all.xlsx')
    limit = excel['贷款额度']
    level=excel['信誉评级']
    lla=0
    llb=0
    llc=0
    for i in range(len(limit)):
        if level[i]=='A':
            lla+=limit[i]
        elif level[i]=='B':
            llb+=limit[i]
        elif level[i]=='C':
            llc+=limit[i]

    excel=pd.read_excel('file/relation.xlsx')
    IR=excel['贷款年利率']
    WR1=excel['信誉评级 A']
    WR2=excel['信誉评级 B']
    WR3=excel['信誉评级 C']

    temp=0
    f1=WR1[0]
    f2=WR2[0]
    f3=WR3[0]
    ir=IR[0]
    for i in range(len(IR)):
        if lla*(1-WR1[i])+llb*(1-WR2[i])+llc*(1-WR3[i])*IR[i] >=temp:
            temp=lla*(1-WR1[i])+llb*(1-WR2[i])+llc*(1-WR3[i])*IR[i]
            f1=WR1[i]
            f2 = WR2[i]
            f3 = WR3[i]
            ir=IR[i]
        print(lla*(1-WR1[i])+llb*(1-WR2[i])+llc*(1-WR3[i])*IR[i])



get_MAI()
#limit_money()
#limit_money()
#change_raw_data()
#get_weight()

#get_p()
#standard_weight()
#dispose_standard()

#calculate_standard()
#calculate_average()
```

```
#valid_rate()stand
#negative_rate()
#per_month_profit()
```

附录 2：Python：task2.py

```
# -*- coding: gb2312 -*-
import pandas as pd
import xlsxwriter
import math

def valid_rate():
    excel_3=pd.read_excel('file_2/销项发票信息.xlsx')
    workbook = xlsxwriter.Workbook('file_2/NO_1.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "销项发票有效发票占比")

    sale_id=excel_3['企业代号']
    sale_state=excel_3['发票状态']
    now_id=sale_id[0]
    valid=0
    invalid=0
    line=1
    for i in range(len(sale_id)):
        if  sale_id[i]==now_id:
            if sale_state[i]=='有效发票':
                valid+=1
            else:
                invalid+=1
        else:
            if(valid+invalid==0):
                print(0)
            else:
                worksheet.write(line,0,valid/(valid+invalid))
                line+=1
                print(valid/(valid+invalid))
            valid=0
            invalid=0
            if sale_state[i]=='有效发票':
                valid=1
            else:
                invalid=1
            now_id = sale_id[i]

        if i == len(sale_id) - 1:
            print(valid/(valid+invalid))
            worksheet.write(line,0,valid/(valid+invalid))
    workbook.close()

def limit_money():
    workbook = xlsxwriter.Workbook('file/limit_money.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0,0,'企业代号')
    worksheet.write(0,1,'贷款额度')

    level_a=0
    level_b=0
    level_c=0
```

```
    #the number of level A/B/C company
    excel=pd.read_excel("企业信息.xlsx")
    level=excel['信誉评级']
    id=excel['企业代号']

    danger_index = excel['风险指数']
    danger_a=0
    danger_b=0
    danger_c=0

    for i in range(len(level)):
        if level[i]=='A':
            level_a+=1
            danger_a+=danger_index[i]
        elif level[i]=='B':
            level_b+=1
            danger_b += danger_index[i]
        elif level[i]=='C':
            level_c+=1
            danger_c += danger_index[i]
    para_all=3*level_a+2*level_b+level_c
    para_a=(3*level_a)/para_all
    para_b=(2*level_b)/para_all
    para_c=level_c/para_all


    line=1
    for i in range(len(danger_index)):
        if danger_index[i] > 0.745:
            worksheet.write(line,0,id[i])
            if level[i]=='A':
                worksheet.write(line,1,para_a*danger_index[i]/danger_a)
            if level[i]=='B':
                worksheet.write(line,1,para_b*danger_index[i]/danger_b)
            if level[i]=='C':
                worksheet.write(line,1,para_c*danger_index[i]/danger_c)
            line+=1

    workbook.close()

def negative_rate():
    excel_3=pd.read_excel('file_2/销项发票信息.xlsx')
    workbook = xlsxwriter.Workbook('file_2/NO_2.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "销项发票非负数发票占比")

    sale_id=excel_3['企业代号']
    sale_state=excel_3['金额']
    now_id=sale_id[0]
    valid=0
    invalid=0
    line=1
    for i in range(len(sale_id)):
        if  sale_id[i]==now_id:
            if sale_state[i]>0:
                valid+=1
            else:
                invalid+=1
        else:
```

```
            if(valid+invalid==0):
                print(0)
            else:
                worksheet.write(line,0,valid/(valid+invalid))
                line+=1
                print(valid/(valid+invalid))
            valid=0
            invalid=0
            if sale_state[i]>0:
                valid=1
            else:
                invalid=1
            now_id=sale_id[i]

        if i == len(sale_id) - 1:
            print(valid/(valid+invalid))
            worksheet.write(line,0,valid/(valid+invalid))
    workbook.close()

def per_month_profit():
    excel_2 = pd.read_excel('file_2/进项发票信息.xlsx')
    excel_3=pd.read_excel('file_2/销项发票信息.xlsx')
    workbook = xlsxwriter.Workbook('file_2/NO_3.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "企业月均利润")

    sale_id=excel_3['企业代号']
    sale_state=excel_3['金额']
    price_tex_sum=excel_2['价税合计']

    now_id=sale_id[0]
    total=0
    line=1
    for i in range(len(sale_id)):
        if  sale_id[i]==now_id:
            total+=sale_state[i]-price_tex_sum[i]
        else:
            print(total)
            worksheet.write(line,0,total)
            line+=1
            total=0
        if i==len(sale_id)-1:
            print(total)
            worksheet.write(line,0,total)
        now_id=sale_id[i]
    workbook.close()

def standard_weight():
    workbook = xlsxwriter.Workbook('file_2/weight 标准化后的数据.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0, 0, "销项发票有效发票占比")
    worksheet.write(0, 1, "销项发票非负数发票占比")
    worksheet.write(0, 2, "企业月均利润")

    excel=pd.read_excel('file_2/data.xlsx')
    min_1=excel['min_1'][0]
    min_2 = excel['min_2'][0]
    min_3 = excel['min_3'][0]
```

```
    max_1=excel['max_1'][0]
    max_2 = excel['max_2'][0]
    max_3 = excel['max_3'][0]

    valid_rate = excel['销项发票有效发票占比']
    nagetive_rate = excel['销项发票非负数发票占比']
    profit = excel['企业月均利润']


    line=1
    for i in range(len(valid_rate)):
        worksheet.write(line, 0, str((valid_rate[i] - min_1) / (max_1 - min_1)))
        worksheet.write(line, 1, str((nagetive_rate[i] - min_2) / (max_2 -
min_2)))
        worksheet.write(line, 2, str((profit[i] - min_3) / (max_3 - min_3)))
        line+=1
    workbook.close()

def get_weight():
    excel = pd.read_excel('file_2/weight 标准化后的数据.xlsx')
    valid = excel['是否违约']
    credit = excel['信誉评级']
    valid_rate = excel['销项发票有效发票占比']
    nagetive_rate = excel['销项发票非负数发票占比']
    profit = excel['企业月均利润']

    E1=0.9485
    E2=0.9950
    E3=0.7974
    E4=0.9978

    E=E1+E2+E3+E4
    k=4
    w1=(1-E1)/(k-E)
    w2 = (1 - E2) / (k - E)
    w3 = (1 - E3) / (k - E)
    w4 = (1 - E4) / (k - E)

    print(w1,w2,w3,w4)

    workbook = xlsxwriter.Workbook('file_2/风险指数.xlsx')
    worksheet = workbook.add_worksheet()
    worksheet.write(0,0,"企业代号")
    line =1
    for i in range(len(valid)):
        if credit[i]=='A':
            credit[i]=1
        elif credit[i]=='B':
            credit[i]=0.75
        elif credit[i]=='C':
            credit[i]=0.5
        elif credit[i]=='D':
            credit[i]=0.25


danger=w1*valid[i]+w2*valid_rate[i]+w3*nagetive_rate[i]+w4*profit[i]+credit
[i]

        worksheet.write(line,0,danger)
```

```
        line+=1
    workbook.close()

get_weight()
#standard_weight()
#per_month_profit()
#valid_rate()
#negative_rate()
```

附录 3：Python：task3.py

```
from numpy import *


def loadDataSet(fileName, delim='\t'):
    fr = open(fileName)
    stringArr = [line.strip().split(delim) for line in fr.readlines()]
    datArr = [list(map(float, line)) for line in stringArr]
    return mat(datArr)


def pca(dataMat, topNfeat=9999999):
    meanVals = mean(dataMat, axis=0)
    meanRemoved = dataMat - meanVals
    covMat = cov(meanRemoved, rowvar=0)
    eigVals, eigVets = linalg.eig(mat(covMat))
    eigValInd = argsort(eigVals)
    eigValInd = eigValInd[:-(topNfeat + 1):-1]
    redEigVects = eigVets[:, eigValInd]
    print(meanRemoved)
    print(redEigVects)
    lowDDatMat = meanRemoved * redEigVects
    reconMat = (lowDDatMat * redEigVects.T) + meanVals
    return lowDDatMat, reconMat


dataMat = loadDataSet('data.txt')
lowDMat, reconMat = pca(dataMat, 1)


def plotPCA(dataMat, reconMat):
    import matplotlib
    import matplotlib.pyplot as plt
    datArr = array(dataMat)
    reconArr = array(reconMat)
    n1 = shape(datArr)[0]
    n2 = shape(reconArr)[0]
    xcord1 = []
    ycord1 = []
    xcord2 = []
    ycord2 = []
    for i in range(n1):
        xcord1.append(datArr[i, 0]);
        ycord1.append(datArr[i, 1])
    for i in range(n2):
        xcord2.append(reconArr[i, 0]);
        ycord2.append(reconArr[i, 1])
    fig = plt.figure()
```

```
    ax = fig.add_subplot(111)
    ax.scatter(xcord1, ycord1, s=90, c='red', marker ='^')
    ax.scatter(xcord2, ycord2, s=50, c='yellow', marker ='o')
    plt.title('PCA')
    plt.show()


plotPCA(dataMat, reconMat)
```

附录 4：Python：task4.py

```
# -*- coding: gb2312 -*-

from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn import svm
from sklearn.metrics import mean_squared_error, mean_absolute_error
import pandas as pd
import numpy as np
import xlsxwriter

#read_excel
data=pd.read_excel('task.xlsx')
A=data['是否违约']
#predict=data['信誉评级']
C=data['销项发票有效发票占比']
D=data['销项发票非负数发票占比']
E=data['企业月均利润']

pre=pd.read_excel('file/change_raw_data.xlsx')
predict=pre['信誉评级']

#转换为 list
#A=np.array(A)
#A=A.tolist()
C=np.array(C)
C=C.tolist()
D=np.array(D)
D=D.tolist()
E=np.array(E)
E=E.tolist()
predict=np.array(predict)
predict=predict.tolist()
for i in range(len(predict)):
    predict[i]*=100
df=pd.DataFrame({'A':A,#特征 1
                 'C':C,
                 'D':D,#特征 4
                 'E':E,#特征 5
                 'predict_result':predict})#预测的结果
X=df.loc[:,['A','C','D','E']]
y=df['predict_result']
clf = svm.SVC()
clf.fit(X, y.astype(int))

#new 为放入的特征
num=0
```

```
data2=pd.read_excel('file_2/weight 标准化后的数据.xlsx')
A2=data2['是否违约']
#predict=data['信誉评级']
C2=data2['销项发票有效发票占比']
D2=data2['销项发票非负数发票占比']
E2=data2['企业月均利润']
workbook = xlsxwriter.Workbook('file_2/预测结果_level.xlsx')
worksheet = workbook.add_worksheet()
worksheet.write(0,0,"信誉评级")
y_pred=[]
line=1
print(len(C2))
for i in range(len(C2)):
    new=[[A2[i],C2[i],D2[i],E2[i]]]

    if(clf.predict(new)==25):
        worksheet.write(line,0,'D')
    elif (clf.predict(new) == 50):
        worksheet.write(line, 0, 'C')
    elif (clf.predict(new) == 75):
        worksheet.write(line, 0, 'B')
    elif (clf.predict(new) == 100):
        worksheet.write(line, 0, 'A')

    #worksheet.write(line, 0, clf.predict(new))
    print(clf.predict(new))
    y_pred.append(clf.predict(new))
    line+=1
print(line)
workbook.close()
```

附录 5：Python：task5.py

```
import numpy as np

# 1.读取数据集
data = np.loadtxt('./2.data', dtype=str, delimiter=',')

a = int(100000000)
AN, BN, CN = 0, 0, 0
AM, BM, CM = 0, 0, 0
for i in range(len(data)):
    if data[i][1] == "A":
        AN += 1
        AM += float(data[i][2])
    elif data[i][1] == "B":
        BN += 1
        BM += float(data[i][2])
    elif data[i][1] == "C":
        CN += 1
        CM += float(data[i][2])

LLa, LLb, LLc = 0, 0, 0
LLaSum, LLbSum, LLcSum = 0, 0, 0
upA, upB, upC = 0, 0, 0

for i in range(len(data)):
    if data[i][1] == "A":
```

```
        LLa = (a * 3 * AN * float(data[i][2])) / ((3 * AN + 2 * BN + CN) * AM)
        if LLa > 1000000:
            LLaSum += 1000000
            upA += LLa - 1000000
            print(data[i][0], 1000000)
        else:
            LLaSum += LLa
            print(data[i][0], LLa)
    elif data[i][1] == "B":
        LLb = (a * 2 * BN * float(data[i][2])) / ((3 * AN + 2 * BN + CN) * BM)
        if LLb > 1000000:
            LLbSum += 1000000
            upB += LLb - 1000000
            print(data[i][0], 1000000)
        else:
            LLbSum += LLb
            print(data[i][0], LLb)
    elif data[i][1] == "C":
        LLc = (a * CN * float(data[i][2])) / ((3 * AN + 2 * BN + CN) * CM)
        if LLc > 1000000:
            LLcSum += 1000000
            upC += LLc - 1000000
            print(data[i][0], 1000000)
        else:
            LLcSum += LLc
            print(data[i][0], LLc)

IR = np.loadtxt('./IR.data', dtype=float, delimiter=',')

WR = np.loadtxt('./WR.data', dtype=float, delimiter=',')

print("------------------------------")

for i in range(len(IR)):
    MAI = IR[i] *
((LLaSum*(1-WR[i][0]))+(LLbSum*(1-WR[i][1]))+((LLcSum+upA+upB+upC)*(1-WR[i]
[2])))
    print(MAI)
```