基于浮动利率的中小微企业信贷模型

摘要

中小微企业规模较小、缺少抵押资产、抗风险能力低，因此如何制定针对中小微企业的信贷策略成为银行的重要工作。本文依据题目所给数据，对中小微企业信贷风险进行量化，并据此建立不同情况下的信贷策略模型。

针对问题一，本文首先从**企业实力**、**上下游企业的稳定度**、**信誉度**、**行业风险** 4 个方面进行考虑，并将贷款企业与上下游企业的关系抽象为**图**，分析上下游企业的实力，据此建立了 10 个量化风险的指标。在此基础上，使用主成分分析法对 10 个指标降维，得到 4 个主成分。再使用熵权法对主成分确定风险权重，最终得到 4 个主成分的权重分别为：0.421、0.165、0.335、0.078，然后通过线性加权得到企业的信贷风险值。对于银行的信贷政策，本文首先通过信用等级、违约记录、信贷风险值判断是否允许企业贷款。对于允许贷款的企业，本文通过构建**企业还款概率矩阵**刻画企业的还款能力，根据附件三给出统计数据，拟合贷款利率和客户流失率的关系式，并通过单目标优化模型确定企业的贷款利率与贷款额度。最后，本文分析了信贷风险值与企业是否有违约记录之间的关系，得到**斯皮尔曼相关系数为 0.881**，表明本文的信贷风险值对企业信誉有良好的刻画能力。

针对问题二，基于第一问中确定的 10 个指标，利用决策树模型，对缺失的企业的违约记录与信用评级信息进行补全。在此基础上，利用问题一中的信贷风险模型，计算得到 302 个企业的信贷风险值。接着，在第一问模型的基础上，对已经合格予以放贷的企业进行分类，根据**自相关系数**的大小，将所有企业分为“周期型企业”和“非周期性企业”，并统计各“**周期性企业**”的周期。对于周期性企业，设定浮动利率，采取“进货期利率下调，贷款额度上调”的优惠策略。通过考虑贷款额度上调对“**企业粘性**”的影响，对客户流失率指标进行修正。最后，求解修正后的单目标规划模型，得到银行年利润下降了 5.89%，客户平均流失率下降了 6.00%。

针对问题三，本文分析了**新冠疫情**和**洪涝灾害**对不同行业中小微企业的影响。对于新冠病毒疫情对企业的影响，本文首先对企业所在行业进行划分，针对原模型进行调整，并基于“**提升银行对不良企业容忍度原则**”、“**专项额度原则**”和“**利率调整原则**”，对银行的信贷策略做出调整。调整后，受疫情负面影响企业的平均利率下降了 3.639%。对于洪涝灾害对企业的影响，本文分析了洪涝灾害对企业冲击的逐级传播关系，根据企业受到的冲击不同，对利率进行了调整，调整后的遭受冲击的企业平均利率降低了 0.160%。

最后，本文对决定企业还款概率的参数进行**灵敏度分析**。通过实验，当参数变化范围不超过 0.4 时，企业平均还款概率的变化范围不超过 0.025。在参数浮动时，所有企业的平均还款概率的波动较小，体现了参数对还款概率的影响并不敏感，模型对于参数的变化是稳定的。

关键词：浮动利率 企业黏性 自相关系数 主成分分析

一、问题重述

1.1 背景资料与条件

在企业与银行的借贷过程中，由于中小微企业的体量较小、缺乏充足的资产用于抵押，因此银行通常通过信贷政策、企业的交易单据信息、贷款对象的上下游企业还款能力等因素对借贷进行决策。企业的信贷过程如图 1 所示，需要对企业进行相关的评估，再据此确定企业的信贷策略。银行通常选择高信誉、信贷等级高的企业进行贷款。除了决定是否向企业进行贷款外，银行还需要根据相关企业的风险评估相关信息对准备贷款的企业进行贷款金额、贷款利率和贷款期限等策略的决策。

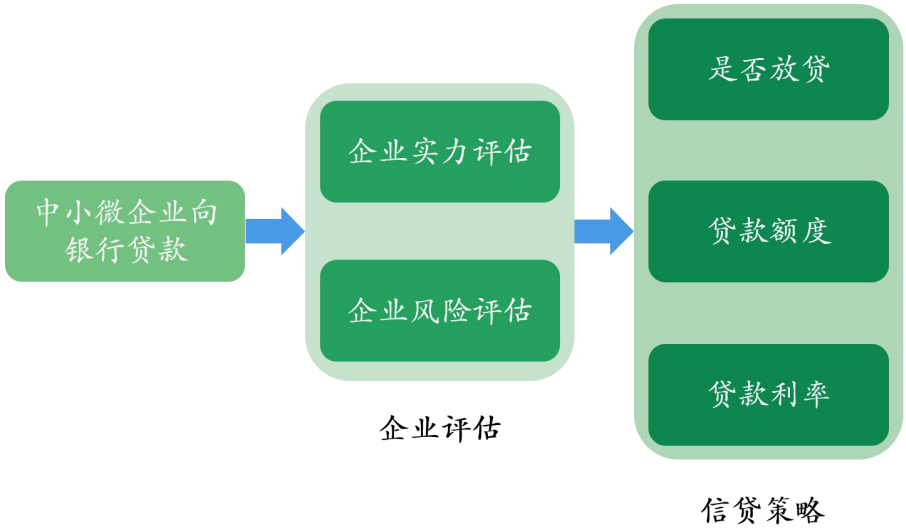


图 1 银行对中小微企业的信贷策略图

1.2 需要解决的问题

某个银行对确定需要放贷企业的贷款额度范围为 10~100 万元；贷款年利率为 4%~15%；贷款的期限为 1 年。

（1）对附件 1 中的 123 家企业进行风险量化分析，并给出在银行当年信贷总额一定时的信贷策略。

（2）基于问题 1，对模型进行改进。对附件 2 中的 302 家企业的信贷风险进行量化评估，同时给出这家银行在年度信贷总额为 1 亿元时对这些企业的信贷策略。

（3）在实际中，企业的生产和最终的经济效益很有可能会受到突发情况的影响。这些因素很有可能会对中小微企业造成影响，不同类型、行业的企业也会受到不同程度的影响。分析附件 2 中各个企业的信贷风险和可能存在的突发因素对各个企业产生的影响，给这家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

1、问题一的分析

首先，为了对 123 家企业的信贷风险进行量化分析，我们将充分挖掘题中所给附件一、三的数据中蕴含的信息，遵循全面性、系统性、针对性原则，从企业实力、供求关系稳定度、信誉度、行业风险四个角度进行考虑，确定出 10 个指标进行量化分析。在此基础上，由于指标过多，考虑使用主成分分析法对这 10 个指标进行降维。对于降维得到的主成分，再使用熵权法确定这些主成分的权值，从而得到企业信贷风险的计算公式。将 123 家企业的数据代入此计算时，最终计算出每家企业的信贷风险得分，并检验模型的合理性。

其次，为了确定银行在年度信贷总额固定时对这些企业的信贷策略，我们首先根据信誉评级、违约记录、信贷风险和年利润等指标，剔除掉不合格企业，对这些企业，不予借贷。

接着，考虑各合格企业的的还款概率。基于企业信贷风险和贷款额度，给出各企业的还款概率量化值，从而形成企业还款概率矩阵。另一方面，根据附件三给出统计数据，我们选取幂函数拟合贷款利率和客户流失率的关系式，在此基础上，以各企业贷款概率和额度为决策变量，建立“银行盈利最大化”的非线性规划模型。

最后，代入银行信贷总额进行求解。

2、问题二的分析

首先，为了对 302 家企业的信贷风险进行量化分析，我们通过对比第一问和第二问所给数据差异，发现第二问所有企业缺少信誉评级指标和违约记录数据，故拟通过决策树算法，，对这些缺失的数据进行预测。将预测出来的数据与附件二所给数据一起，采用第一问的信贷风险量化模型，计算得到 302 家企业的信贷风险得分。

其次，通过分析 302 家企业的年利润波动曲线，发现相当一部分企业的累计年利润随时间呈周期性波动。考虑到这类企业的进货、卖货期间对于借贷金额的需求存在较大的差异，故在第一问的基础上，加入对周期性企业的“优惠政策”。

最后，在第一问“信贷策略模型”的基础上，对已经合格予以放贷的企业进行分类，将所有企业分为“周期型企业”和“非周期性企业”，并统计各“周期性企业”的周期。对于周期性企业，采取一定的优惠策略。取各企业周期的一半作为优惠期时长，在优惠期和非优惠期根据不同的信贷策略，设定浮动利率，重新修正第一问的非线性规划模型，最终求出各非周期性企业对应的利率和贷款额度，以及周期性企业的浮动利率和贷款额度。

3、问题三的分析

首先，我们选取两个突发因素为新冠病毒疫情和旱涝灾害，分别考察这两个因素对银行信贷策略的影响。

一方面，对于新冠病毒疫情，我们在查阅大量文献的基础上，对题中所给 302 家企业，按照疫情对其的影响进行分类。接着，根据“提升银行对不良企业容忍度原则”、“专项额度原则”和“利率调整原则”，对银行的信贷策略做出调整。从而得到新冠病毒疫情影响下的银行信贷策略。

另一方面，对于旱涝灾害，我们首先提取出可能受到旱涝灾害直接或间接影响的企业，并将这些企业按照“上下游”关系分层。接着，我们将“产业链”类比“食物链”，结合能量传递逐级递减原理，计算各层企业的受冲击指数。在此基础上，依据冲击指数和企业的信贷风险值，对企业的利率进行调整。

最后，代入具体数值，求解出最终的银行信贷调整策略。

三、模型假设

(1) 题中所给发票数据为所涉及企业的全部发票数据。

(2)“周期型企业”的进货期长度近似为其周期的一半。

(3) 银行在决策最优信贷策略时，只考虑使其自身年利润、信贷风险与突发事件的影响，不考虑其他因素影响。

四、符号说明

符号	说明
$Credit_i$	企业的信誉度
P_k	月均利润
S_i	上下游企业实力
$P_{i,k}$	总资金流量
$\overline{n}_{out,i}$	月内日均负项销项发票数
ρ_s	行业 s 的风险系数
$\overline{n}_{in,i}$	月内日均负项入项发票数
z_i	月均作废发票数
ε_{up}	贷款企业对上游企业的依赖程度
ε_{down}	贷款企业对下游企业的依赖程度

五、模型的建立与求解

5.1 企业信贷风险的分析与策略

5.1.1 企业信贷风险的量化分析

银行对企业进行借贷时，需要对企业的实力、信用等因素进行量化分析，根据中小微企业能否按期还款确定相关策略。由于本文需要通过相关中小微企业的大量数据进行分析，本文将对相关的指标进行针对性分析，对各指标的关系进行系统性分析。此外，由于所给数据类型的限制，本文还将考虑各个指标的可度量性。

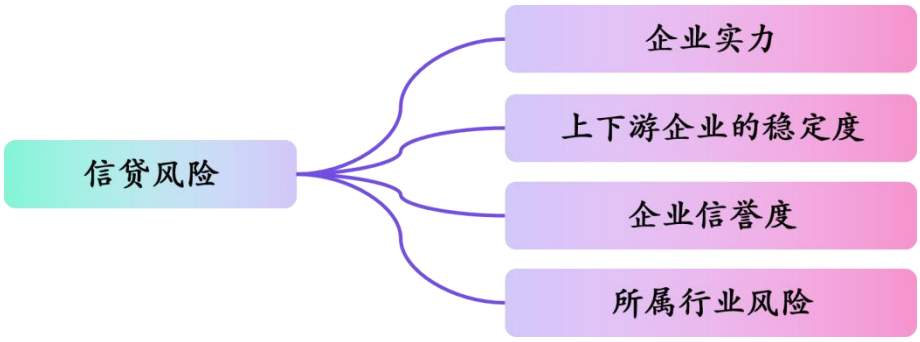


图 2 企业信贷风险评估角度分析图

如图 2 所示，本文将从企业实力、贷款企业上下游企业的稳定度、企业信誉度、企业所属行业风险对企业的信贷风险进行评估。

• 数据预处理

对于“附件 1”中企业的发票单据数据，本文首先对数据进行预处理。在“进项发票信息”表和“销项发票信息”表中，发票的状态分为有效发票和作废发票两种状态。有效发票为企业在进行正常交易活动时所开具的发票；作废发票是企业在为交易活动开具这张发票之后，因故取消该项交易，使发票作废。因此，本文在统计进项发票、销项发票金额信息时，对表中的作废发票进行剔除，再对剔除后的数据进行分析。

• 风险量化的指标

对于经过预处理之后的数据，首先对企业的实力进行分析。

（一）企业实力

对于某家中小微企业，在销项发票中出现负数发票可以反应企业的销售实力。因为负数发票是企业在为交易活动开局发票之后，由于购货方因故退货、退款所开户的发票，这多数是由于企业本身实力不足造成的。大量的退货、退款所产生的负数发票可以反映出企业自身的实力。根据企业相关数据，得到企业在一定时间（如一个月）范围内第 i 家企业的月均负数销项发票数为：

$$\overline{n}_{out,i} = \frac{1}{k} \cdot \sum_{j=1}^{k} n_{out,i,j} \ , \tag{1}$$

其中 $n_{out,i,j}$ 表示第 i 家企业第 j 月的负数销项发票数，k 表示这一段时间包含的月数。

此外，第 i 家企业在第 k 月的月净利润 $P_{i,k}$ 也是衡量企业盈利实力的重要指标。进项发票和销项发票分别表示企业进货时销售方开具的发票和销售时购货方开具的发票。二者额度的差值表示企业在这一段时间内的盈利。企业在第 k 月的月净利润 $P_{i,j}$ 可表示为：

$$P_{i,k} = \sum_{i=1}^{n} n_{in,i,k} - \sum_{i=1}^{n} n_{out,i,k} \ , \tag{2}$$

其中 $n_{in,i,k}$ 、$n_{out,i,k}$ 分别表示第 i 家企业第 k 月的进项发票金额和销项发票金额。则企业的月均利润为：

$$P_k = \frac{1}{K} \sum_{k=1}^{K} P_{i,k} \ . \tag{3}$$

（二）上下游企业的稳定度

如图 3 所示，由于上游、下游企业存在向多家企业供货、销货的行为，本文将其资金流向关系抽象为图。在衡量企业的实力时，上下游企业的实力也是影响企业实力的一个关键指标。本文通过贷款企业上游企业的销售商数量反映上游企业的供货水平与实力；通过贷款企业下游企业的供货商数量反应下游企业的物资消耗水平与实力。贷款企业的上下游企业实力为：

$$S_i = \sum n_{up} + \sum n_{down} \ . \tag{4}$$

其中，n_{up}, n_{dowm} 分别表示上游、下游企业的实力，即上游、下游企业的供货、销货商数量。

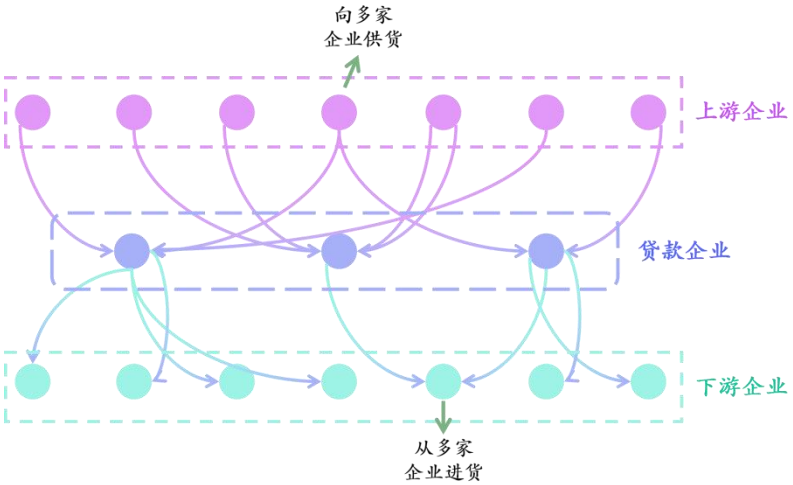


图 3 上下游企业资金流向关系图

此外，企业交易进项、销项的总金额也是衡量企业资金流量大小与实力的重要指标。第 i 家企业第 k 月的总资金流量为：

$$P_{i,k}=\sum_{i=1}^{n}n_{in,i,k}+\sum_{i=1}^{n}n_{out,i,k}\,. \tag{5}$$

当银行在向中小微企业进行借贷时，上下游企业供求关系的稳定性也将影响企业的还款能力。

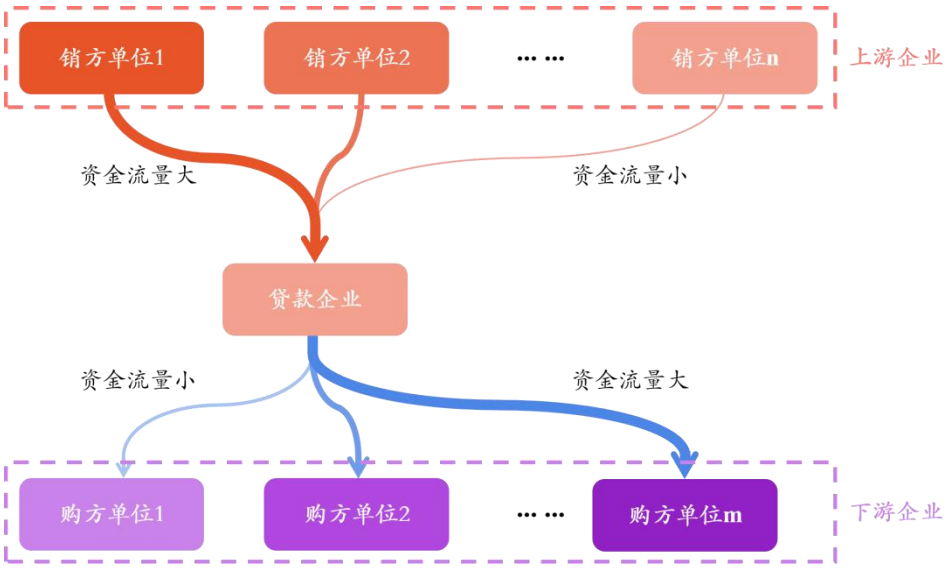


图 4 企业上下游的供应关系图

本文将企业的上下游供应关系抽象为图。图 4 表示的是企业上下游的供应关系图，资金流线条越粗表示企业间的资金流越大，反之则表示资金流越小。图 4 中的企业间资金流向可以反应企业与上下游企业之间的交易关系。如图 4 所示，当企业在进货时过分依赖某一家厂商的货源时，很有可能在这家上游企业资金链断裂时不能继续生产，进而导致该企业陷入需求紧张的危机；当在下游企业在无力支付相关费用或需求大幅减少时销售链断裂，导致贷款企业库存累积、资金无法回流，进而导致无法偿还银行债务。

根据上述分析，如果在资金流通的过程中如果过分依赖于某一或某几个上、下游企业的资金流入、流出，就很有可能在这几家上、下游企业出现特殊风险时，贷款企业陷入危机。企业资金过分依赖某几家上、下游企业表现为资金分布的不均衡性。因此本文通过每个月上下游企业入、销项金额的样本方差 ε_{up} 、 ε_{down} 分别表示贷款企业对上、下游企业的依赖程度，且：

$$\varepsilon_{up,i}=\frac{1}{m_{in}-1}\sum_{j=1}^{m_{in}}(e_{up,i,j}-\bar{e}_{up,i})^2\,, \tag{6}$$

$$\varepsilon_{down}=\frac{1}{m_{down}-1}\sum_{j=1}^{m_{down}}(e_{down,i,j}-\bar{e}_{down,i})^2\,. \tag{7}$$

其中 $e_{up,i,j}$ 、 $e_{down,i,j}$ 分别表示上游企业 j 流向第 i 家贷款企业的资金记录值与第 i 家贷款企业流向下游企业 j 的资金记录值，其中 m_{in} 、 m_{out} 分别是上游资金流向贷款企业与贷款企业流向下游企业资金的记录条数。

对贷款企业的上游企业质量进行评估。本文对贷款企业的上游企业的负数发票进统计。根据企业相关数据，得到企业在一个月的时间范围内第i家企业的日均负项入项发票数为：

$$\bar{n}_{i,j}=\frac{1}{k}\cdot\sum_{j=1}^{k}n_{in,i,j}\,. \tag{8}$$

对贷款企业的下游企业信誉进行评估。由于作废发票是在为交易活动开具发票后，因故取消了该项交易而导致作废的。因此，为下游企业开具发票作废的数目可以反映下游企业的交易信誉。当下游企业有较高信誉度时，贷款企业i所承担的风险较小。贷款企业的月均作废发票数为：

$$z_i=\frac{1}{k}\cdot\sum_{j=1}^{k}z_{i,j}\,. \tag{9}$$

其中，$z_{i,j}$表示第j月贷款企业i为下游企业开具发票作废数目。

（三）企业信誉度

根据“附件 1”中的企业信誉评级结果，对企业的信誉等级进行赋分。各个企业对应信用等级的赋分$Degree_i$可以表示为：

$$Degree_i=\begin{cases}0,\ \text{等级}D\\1,\ \text{等级}C\\2,\ \text{等级}B\\3,\ \text{等级}A\end{cases}. \tag{10}$$

由于部分企业存在违约记录，对公式（10）的信用等级进行修正，得到修正后企业的信誉度$Credit_i$表达式为：

$$Credit_i=Dgree_i\cdot\lambda_i\,, \tag{11}$$

其中，当贷款企业有违约记录时$\lambda_i=0$；当贷款企业无违约记录时$\lambda_i=1$。

（四）所属行业风险

由于企业所在行业的不同会造成资金流量、抗风险能力的不同。因此，在贷款过程中，银行还需要对贷款企业所在的行业风险进行评估。根据“附件 1”中的企业名称，本文将相关企业分类为：商业、技术行业、工业、商业、邮政物流业、生活服务业、文化传媒业、农林牧渔业、个体户 9 个行业类别。通过各个行业的信誉度均值得到行业s的风险系数ρ_s为：

$$\begin{aligned}\rho_s&=\frac{1}{n_s}\sum_{i=1}^{n_s}(\max\{Degree_j\}-Degree_i)\\&=\frac{1}{n_s}\sum_{i=1}^{n_s}(3-Degree_i)\end{aligned}\,. \tag{12}$$

其中n_s表示 123 家企业中行业s的企业数量。

通过从企业实力、贷款企业上下游企业的稳定度、企业信誉度、企业所属行业风险对企业的信贷风险进行评估。得到的 10 个参考指标如表 1 所示。

表 1 企业信贷风险的评估参考指标表

统计量	企业的信誉度	月均利润	上下游企业实力	总资金流量
符号	$Credit_i$	P_k	S_i	$P_{i,k}$
统计量	月内日均负项销项发票数	行业 s 的风险系数	月内日均负项入项发票数	月均作废发票数
符号	$\overline{n}_{out,i}$	ρ_s	$\overline{n}_{in,i}$	z_i
统计量	贷款企业对上游企业的依赖程度		贷款企业对下游企业的依赖程度	
符号	ε_{up}		ε_{down}	

根据表 1 中确定的相关指标，首先按照公式（13）对所有指标进行归一化处理：

$$\hat{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}. \tag{13}$$

对处理后的数据利用主成分分析法对数据进行降维，以便于对数据的求解。通过 SPSS 对上述指标进行主成分分析。首先进行 KMO 检验，得到 KMO 检验系数为 0.619>0.5，其显著性为 0.000<0.05，适合进行主成分分析。通过主成分分析**[1]** 得到了 4 个主成分为：

$$\begin{aligned} X_1 = {} & 0.124z_i + 0.672P_k + 0.887S_i - 0.990P_{i,k} + 0.251\overline{n}_{in,i} + 0.964\varepsilon_{up} \\ & + 0.459\varepsilon_{down} - 0.234Credit_i - 0.050\rho_s \end{aligned}, \tag{14}$$

$$\begin{aligned} X_2 = {} & 0.595z_i - 0.660P_k + 0.297S_i + 0.104P_{i,k} + 0.551\overline{n}_{in,i} - 0.245\varepsilon_{up} \\ & + 0.532\varepsilon_{down} - 0.272Credit_i - 0.021\rho_s \end{aligned}, \tag{15}$$

$$\begin{aligned} X_3 = {} & -0.348z_i - 0.283P_k + 0.305S_i + 0.026P_{i,k} - 0.547\overline{n}_{in,i} - 0.053\varepsilon_{up} \\ & + 0.655\varepsilon_{down} + 0.465Credit_i + 0.305\rho_s \end{aligned}, \tag{16}$$

$$\begin{aligned} X_4 = {} & 0.418z_i + 0.109P_k - 0.058S_i - 0.060P_{i,k} - 0.249\overline{n}_{in,i} + 0.067\varepsilon_{up} \\ & - 0.215\varepsilon_{down} + 0.531Credit_i + 0.754\rho_s \end{aligned}. \tag{17}$$

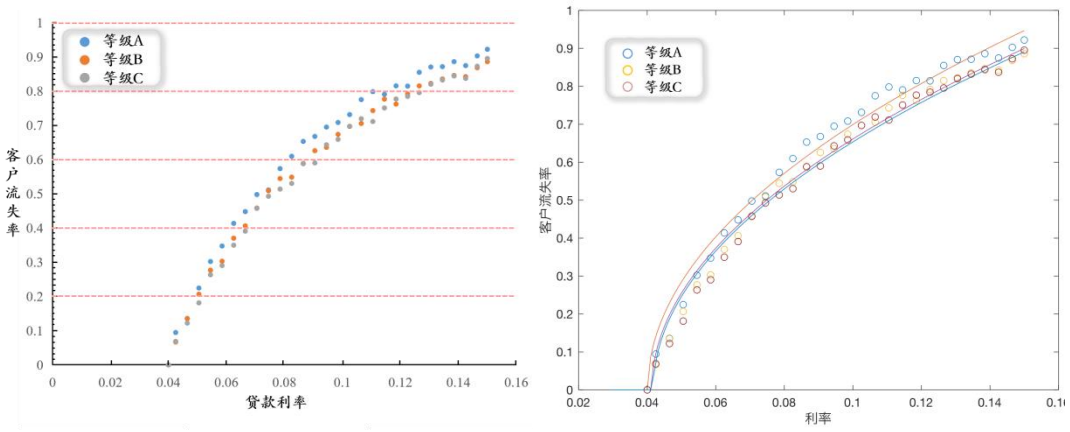
对提取出的因子，本文通过熵权法对 4 个因子确定权重。由于熵权法适用于因变量与自变量正相关的情况，本文首先对于与风险负相关的因子进行正向化。即将与风险负相关的因子 X_i 转化为 $X_i' = -X_i$。对于经过转化后的因子通过熵权法确定权重。通过 Python 计算，得到四个指标所对应的权重分别为：0.421，0.165，0.335，0.078。得到企业信贷风险的量化值为：

$$risk_i = 0.421X_1 + 0.165X_2 + 0.335X_3 + 0.078X. \tag{18}$$

5.1.2 企业的信贷策略

• 模型的准备

在银行的利率不断调整过程中，银行的贷款客户可能会因为利率的提高而流失。“附件 3”中的数据反映了银行贷款年利率与客户流失率之间的离散关系。如图 5(a)所示，在客户信用等级不同时，银行的客户流失率与企业贷款年利率之间有较强的相关性。



(a) (b)

图 5 客户流失率与企业贷款年利率关系图

根据图 5(a)，在贷款利率较小时，客户流失率随贷款利率变化较大；在在贷款利率偏大时，客户流失率随贷款利率变化较小。因此适合采用类幂函数对曲线进行拟合。由于银行贷款的年利率范围为 4%—15%，因此利率的取值必须大于等于 4%。设出如下模型对曲线进行拟合：

$$\alpha(\eta) = K_1 \cdot (\eta - 0.04)^{\frac{1}{2}}. \tag{19}$$

利用最小二乘法，通过 MATLAB 求解得到用户等级分别为 A、B、C 时，客户流失率与银行利率之间的函数关系如公式（20）、（21）、（23）所示，拟合曲线如图 5(b)所示。

$$\alpha(\eta) = 2.856 \cdot (\eta - 0.04)^{\frac{1}{2}}, \tag{20}$$

$$\alpha(\eta) = 2.728 \cdot (\eta - 0.04)^{\frac{1}{2}}, \tag{21}$$

$$\alpha(\eta) = 2.707 \cdot (\eta - 0.04)^{\frac{1}{2}}. \tag{22}$$

• 银行是否对企业放贷

银行在面对企业的贷款需求时，首先需要确定企业的信誉等级。由于信用为 D 的且有将会为银行带来较大风险，因此，原则上银行对信誉等级为 D 的企业不予放贷。如图 6 所示，本文首先将信誉等级为 D 的企业筛除，不予放贷。同样地，对于有失信记录的企业，银行同样不予放贷。

此外，本文还将通过 5.1.1 节中确定的企业风险等级对是否放贷进行筛选。如图 6 所示，首先选取合适的阈值 $r_0 = \min\{risk_i \mid Degree(i) = 0 或 \lambda_i = 0\}$ 作为判断的准则，若第 i 家企业的风险 $risk_i > r_0$，则不允许该企业进行贷款；否则，则允许该企业进行贷款。

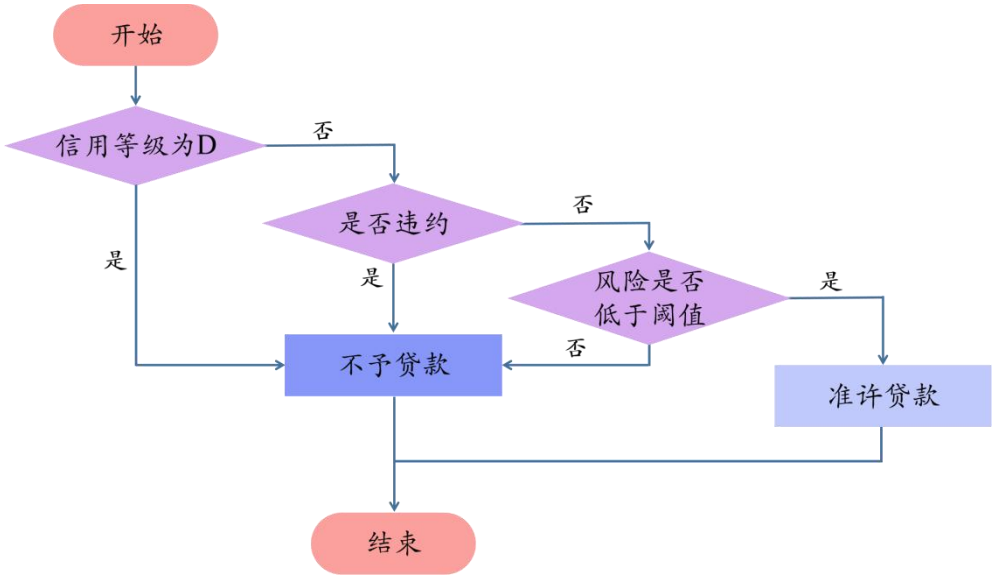


图 6 银行是否对企业放贷决策流程图

• 放贷利率与贷款额度决策

（一）基于信誉分级的还款概率矩阵模型

对于已经剔除过不予放贷的企业后的剩余企业，本文将建立“**基于信誉分级的还款概率矩阵模型**”研究如何确定合适的放贷利率和贷款额度。

在现实生活中，对于企业的贷款额度，银行给予企业的贷款额度通常是金额为整数的数额且贷款金额的范围为 10~100 万元。因此，本文建立的企业贷款模型金额为分级贷款(即 10,20, ... ,90,100 万元)，对应的分级符号用 $u_1, u_2, ..., u_{10}$ 表示(分别对应 10,20, ... ,90,100 万元的额度)。

首先建立**企业还款概率矩阵** $P_{m\times 10}$（m 表示银行决定放贷的企业数)。在矩阵 P 中，第 i 行第 j 列的元素表示第 i 家企业在贷款金额为 u_j 时还款概率 $P_{i,j}$。因为企业的利润决定了企业的还款实力，本文认为企业的还款概率 $P_{i,j}(x)$ 是企业年利润 x 的函数。

下面定义企业还款概率 $P_{i,j}(x)$。对信用等级为 s 的贷款企业，当企业年度贷款额度为 u_j、企业年利润为 x 时，若二者比值 $u_j / x \geq \omega_s$，则该条件下的企业还款概率：

$$P_{i,j}(x) = p_{\max}^{(s)}, \tag{23}$$

其中 $p_{\max}^{(s)}$ 表示信用等级为 s 的贷款企业的最大还款概率。在企业年度贷款额度与企业年利润的比值为 $u_j / x < \omega_s$ 时，本文构建改造后的 Sigmoid 函数对企业还款概率 $P_{i,j}(x)$ 进行描述：

$$P_{i,j}(x) = \frac{a_s}{1+\exp(u_j - \omega_s \cdot x)}. \tag{24}$$

其中 a_s 为需要确定的参数。公式（24）反应了企业还款概率 $P_{i,j}(x)$ 随着贷款额度 u_j 的增加而减小，随企业年利润为 x 增加而增加的规律。

下面进行参数的确定。由于 A、B、C 三种信誉类型企业的信誉递减，相同年利润下的还款意愿也随之递减，因此本文令 $\omega_A = 0.8, \omega_B = 1.0, \omega_C = 1.2$。不同信誉等级企业的的贷款企业的最大还款概率 $p_{\max}^{(s)}$ 也不同，本文 $p_{\max}^{(A)} = 1.0, p_{\max}^{(B)} = 0.9, p_{\max}^{(C)} = 0.8$。

此外，还需要确定不同信誉等级的参数 a_s。为确保函数在 $u_i / x = \omega_s$ 处的连续性，本文令：

$$\lim_{\omega_s x \to u_j^+} P_{i,j}(x) = \lim_{\omega_s x \to u_j^-} P_{i,j}(x)\,, \tag{25}$$

即：

$$p_{\max}^{(s)} = \frac{a_s}{1+\exp(u_j - \omega_s \cdot x)}. \tag{26}$$

根据公式（25）、（26），得到参数 $a_A = 2, a_B = 1.8, a_C = 1.6$。$P_{i,j}(x)$ 的相关参数均得到确定，因此可以由企业的年利润与贷款额度确定其还款概率。

（二）银行信贷策略模型

本文将根据企业还款概率矩阵 P 建立单目标优化模型。银行的收益期望由 2 部分决定，一方面是银行的利息盈利 $u_j \cdot \eta_i$(其中 η_i 为利率)，另一方面是由于坏账(企业未按时归还贷款)导致的亏损金额 u_j。根据企业还款概率 $P_{i,j}(x)$，得到银行的收益期望：

$$E = \sum_{i=1}^{m} [P_{i,j} \cdot u_j \cdot \eta_i - (1 - P_{i,j}) \cdot u_j] \cdot (1 - \alpha_i)\,, \tag{27}$$

其中 α_i 表示公式（20）、（21）、（22）中的客户流失率。在企业 i 的风险为 r_i 时，确定单目标优化模型：

$$\max \sum_{i=1}^{m}[P_{i,j}\cdot u_j\cdot \eta_i-(1-P_{i,j})\cdot u_j]\cdot(1-\alpha_i)$$
$$s.t.\begin{cases}\sum u_j \le U \\ 100{,}000 \le u_j \le 1{,}000{,}000,\ j=1,2,...,10 \\ 4\% \le \eta_i \le 15\%,\ i=1,2,...,m\end{cases} \quad (28)$$

（三）银行信贷策略模型的求解

为对模型进行初步验证，本文将银行的年度信贷总额固定为 5000 万元，通过 MATLAB 对公式（28）的单目标优化方程组进行求解。得到企业还款概率矩阵 $P_{96\times10}$ 如图 7 所示(部分行省略)，其中第 i 家企业在贷款金额为 u_j 时还款概率为 $P_{i,j}$。图 7 中，矩阵元素的颜色从红到绿反应了元素数值大小从小到大，表示企业还款概率的增加，风险的减少。

0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90
0.80	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.80	0.80	0.80	0.80	0.80	0.0	0.80	0.80	0.80	0.80
0.80	0.80	0.80	0.80	0.80	0.80	0.04	0.00	0.00	0.00
1.00	1.00	1.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
... ...	... ...	... ...	... ...	... ...	... ...	... ...	... ...	... ...	... ...
0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.01	0.00	0.00
1.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.90	0.90	0.61	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.90	0.90	0.90	0.90	0.90	0.17	0.00	0.00	0.00	0.00
0.90	0.90	0.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00

图 7 企业还款能力矩阵可视化图

得到在上述条件下银行所投资企业的年利率与贷款额度统计表如表 2 所示。

表 2 企业的年利率与贷款额度统计表

企业代号	E1	E2	E3	... ...	E104	E105	E106	E110
利率	0.09968	0.14985	0.07133	... ...	0.06529	0.06686	0.06077	0.06626
贷款额度	10 万	10 万	100 万	... ...	20 万	120 万	90 万	10 万

5.1.3 模型的分析与检验

根据主成分分析，本文得到了经过数据降维后的 4 个主成分 X_1,X_2,X_3,X_4。在 4 个因子中，指标系数绝对值较大的指标对该因子的影响较大，因子也对该指标有较强的表示能力。在因子 X_1 中，$P_k,S_i,P_{i,k}$ 项的系数绝对值较大，主要对月

均利润、企业收入稳定程度、总资金流量等因素有较强的表示能力，主要可以反映企业的实力。类似地，因子 X_2 中 $Credit_i$ 所占权重较高，对企业的信誉度有较好的表示能力；因子 X_3 中 ε_{down} 所占权重较高，可以反映贷款企业对上下游企业的依赖性；因子 X_4 中 ρ_s 所占权重较高，可以反映贷款企业的行业风险系数。

根据上述分析，可以看出经过主成分分析后的 4 个因子可以在一定程度上反映图 2 所示的 4 个影响企业信贷风险的主要方面，具有一定的现实内涵与可解释性。

根据得到的结果对信贷风险量化分析模型进行检验。首先将各家企业的风险值进行归一化处理：

$$r_i = \frac{risk_i}{\max\{risk_j\}}. \tag{29}$$

下面检验归一化风险 r_i 与该企业是否有违约记录 σ_i 进行相关性分析。计算风险与是否违约之间的斯皮尔曼相关系数：

$$p = \frac{\sum_i (r_i - \bar{r})(\sigma_i - \bar{\sigma})}{\sqrt{\sum_i (r_i - \bar{r})^2 (\sigma_i - \bar{\sigma})^2}}. \tag{30}$$

经过求解，得到斯皮尔曼相关系数为 0.881，表明本文确定的相关性系数与企业是否违约可以呈现良好的相关性，对企业的信誉、风险有良好的刻画能力。

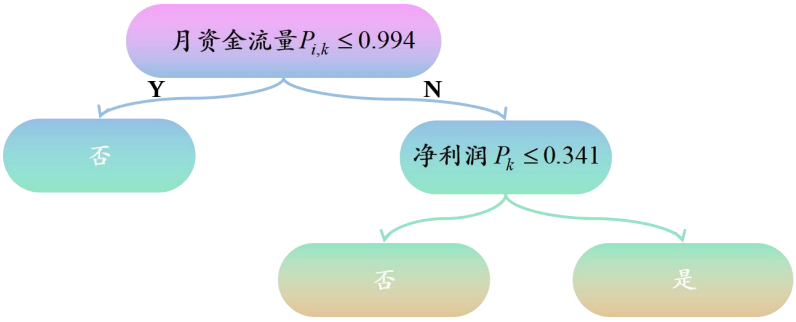
5.2 无信誉记录企业的信贷策略模型

5.2.1 企业信贷数据的量化

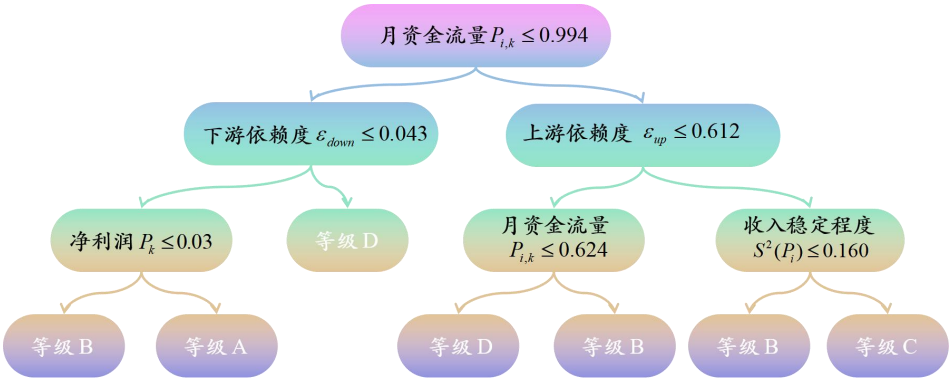
在“附件 2”中的 302 家企业信息表中，缺少企业信用等级的数据，无法确定企业信用等级、违约记录的相关数据。本文利用 5.1 节中得到的 10 个指标，通过决策树模型得到缺失的数据。

表 1 中列举了本文提出的 10 个衡量企业信用等级的指标。由于信用等级、违约记录的缺失，本文选取月均利润 P_k、企业收入稳定程度 $S^2(P_i)$、月资金流量 $P_{i,k}$、月内日均负项销项发票数 $\overline{n}_{out,i}$、月内日均负项入项发票数 $\overline{n}_{in,i}$、月均作废发票数 z_i、贷款企业对上游企业的依赖程度 ε_{up}、贷款企业对下游企业的依赖程度 ε_{down} 8 个指标作为预测企业信用数据的输入数据。

根据公式（13）对数据进行归一化处理，用“附件 1”中归一化处理后的数据对决策树模型进行求解。通过 Python 求解，得到基于上述 8 个因素的决策树如图 8 所示。该模型在“附件 1”中的数据集所划分的训练集上得到 74.8%的准确率，在 78.8%的准确率，在验证集上得到 70.8%的准确率。结果过训练后的模型将预测“附件 2”中的缺失数据。



(*a*)



(*b*)

图 8 经过“附件 1”训练后的是否违约、信誉等级判断决策树

根据图 8 所示的决策树，得到“附件 2”中 302 家企业的预测风险评级如表 3 所示。

表 3 “附件 2”中部分企业的预测风险评级统计表

企业	E124	E125	... ...	E423	E424	E425
信用评级	B	B	... ...	D	C	B
是否违约	否	否	... ...	是	是	是

对于得到的风险评级，根据 5.1 中的风险指标计算方法确定表 1 中各个指标的数值。

5.2.2 信贷策略的确定

与 5.1 节类似，在“附件 2”中 302 家企业的基础上根据表 2 所确定的企业评级结果对信誉不良的企业进行剔除。在 5.1 中模型的基础上进行模型的优化、改进。

• 资金周期性流动与浮动利率

在某些行业中，部分企业在进、销货时具有一定的周期性，因此这些企业的月利润变化曲线将呈现周期性波动。银行将根据企业资金流动的周期性变化规律

动态调整企业的利率[5]。例如，在某一段时间内，企业处于进货期，银行可以适当下调利率；在另一段时间内，企业处于进货期，银行可以适当提高企业的贷款利率。在周期性动态调整利率的过程中，既可以实行在较高利率时提高银行的收益，也可以在银行缺乏资金时挽留住企业客户，该浮动利率模式有利于企业的长期盈利。

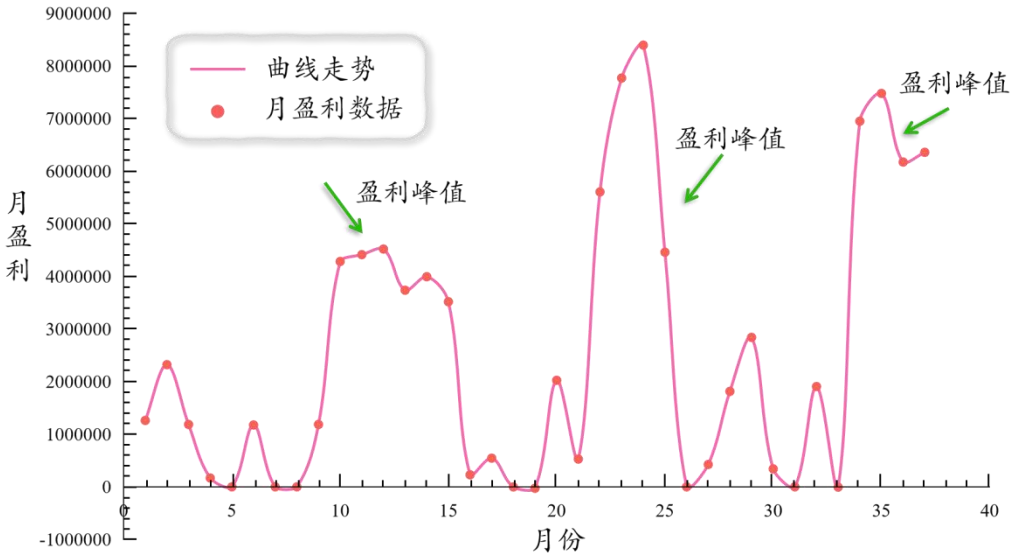


图 9 E185 企业资金周期性流动示意图

下面对资金周期性流动企业进行筛选与挖掘。如图 9 所示，E185 企业(*** 生态魔芋有限公司)，从 2017 年 1 月至 2020 年 1 月期间，共计 37 个月的盈利数据反应了企业的月盈利的周期性波动情况。大致在每年的年底，企业的月销售额达到峰值(图中用绿色箭头标注)，适当地提高利率可以有利于银行的收益；在每年的 6、7 月份，企业的月利润到达低谷，甚至在 2018 年的 6、7 月份出现负增长，适当降低利率有利于稳定银行和企业之间的长期关系，也有利于企业本身发展。

• 筛选周期性企业

下面研究企业是否具有周期性的资金流动规律。对所研究的企业月盈利收益，选取 $P_{1k}=[p_1,p_2,...,p_{12-k}]$， $P_{2k}=[p_{1+k},p_{2+k},...,p_{12}]$2 个序列进行分析，其中 p_i 表示第 i 月的企业净利润。计算这 2 个序列的自相关系数(Pearson 相关系数)作为两个序列之间的相似度：

$$r=\frac{\sum_{i=1}^{n}(P_{1k,i}-\overline{P}_{1k})(P_{2k,i}-\overline{P}_{2k})}{\sqrt{\sum_{i=1}^{n}(P_{1k,i}-\overline{P}_{1k})^2}\sqrt{\sum_{i=1}^{n}(P_{2k,i}-\overline{P}_{2k})^2}} \tag{31}$$

以 $k=2$ 为例，此时选取 $P_1=[p_1,p_2,...,p_{10}]$， $P_2=[p_3,p_4,...,p_{12}]$，计算两序列的相关系数作为其相似度。

本文中 k 的取值为 1,2,...,6,。通过对 k 的取值进行遍历，若两序列的相似度 $r>r_0(r_0=0.6)$时，认为该企业的资金流动呈周期性波动。在一轮遍历中，当 r 取

得最大值且$r>0.6$时，认为该企业的资金流动周期$T=k$；若经过一轮遍历，r均不超过0.6，则该企业的资金流动没有周期性。

在资金周期性流动的企业生产经营过程中，一个资金流动周期内生产和经营的时间通常各占周期的一半，本文认为在生产时期内的时间是企业最缺乏资金的时间，需要在利率浮动上给予一定的优惠。因此本文认为企业的优惠期时长为：

$$T_1=\frac{1}{2}T. \tag{32}$$

一年内其余的时间均为企业的非优惠期，记为T_2。

• 浮动利率对企业黏性的影响

当利率浮动时，会对企业的客户流失率产生影响。在不实行浮动利率策略时，客户流失率与利率的关系为公式（20）、（21）、（22）中所示的函数关系。定义在贷款策略发生变化后的客户流失率：

$$\alpha'=\alpha(\eta)\cdot(1-S), \tag{33}$$

其中S表示在银行实行新策略后的企业黏性。

下面对企业黏性S进行量化。

$$S=\frac{\beta'-1}{2}. \tag{34}$$

其中，$100{,}000\le u'_{down}\le 300{,}000$，

$$\beta'=\frac{u'_{down}}{u_{down}}. \tag{35}$$

其中，u_{down}，u'_{down}分别是上调前、后的企业贷款额度下限，这里$\beta'=3/2$。

• 模型的优化：优惠放贷决策模型

对于资金流量呈周期性波动的m_0家企业，银行在优惠期内的收益期望和非优惠期内的收益期望分别为：

$$E_1=T_1\cdot[\sum_{i=1}^{m_0}[P_{i,j}\cdot u_j\cdot\eta'_i-(1-P_{i,j})\cdot u_j]\cdot(1-\alpha'_i)], \tag{36}$$

$$E_2=T_2\cdot[\sum_{i=1}^{m_0}[P_{i,j}\cdot u_j\cdot\eta_i-(1-P_{i,j})\cdot u_j]\cdot(1-\alpha'_i)], \tag{37}$$

其中T_1,T_2分别为非优惠期时长和优惠期时长。

对于剩余的$m-m_0$家资金流量不呈周期性波动的企业，银行对其收益的期望为：

$$E_3 = \sum_{i=m_0+1}^{m-m_0} [P_{i,j} \cdot u_j \cdot \eta_i - (1 - P_{i,j}) \cdot u_j] \cdot (1 - \alpha_i) . \tag{38}$$

为最大化银行利润，本文根据上述利益期望建立企业利润优化模型：

$$\max E_1 + E_2 + E_3$$
$$s.t. \begin{cases} \sum u_j \le U \\ \eta_i' = \alpha' \eta_i \\ 150{,}000 \le u_j \le 1{,}000{,}000，\ j \in \{\text{周期性企业}\} \\ 100{,}000 \le u_j \le 1{,}000{,}000，\ j \notin \{\text{周期性企业}\} \\ 4\% \le \eta_i \le 15\%，\ i = 1,2,...,m \end{cases} . \tag{39}$$

通过 MATLAB 求解该优化模型，得到“附件 2”中的各个企业在盈利、非盈利周期的利率如表 4 所示。通过对比加入“优惠政策”后的银行信贷策略和之前的银行信贷策略，我们计算得到了修正后的银行预期年收益比原来下降了 5.89%，这说明采用“浮动利率”策略让利给企业，确实会对银行造成一定的损失。但是修正原模型后，该银行对应的各企业平均流失率也下降了 6.00%。这证明了抬高贷款额度和采用“浮动利率”策略，确实对提高企业“忠诚度”具有一定的影响。综合来看，采用“对周期性企业浮动利率策略”，虽然可能短期内使银行蒙受一定的损失，但却将一定程度上提升“企业黏性”，从长远来看，这项策略反而能帮助银行更好地进行盈利。

表 4 各个企业在盈利、非盈利周期的利率统计表

企业代号	E124	E125	E126	... ...	E406	E407	E419	E420
利率(周期)	0.090	0.135	0.135	... ...	0.136	0.135	0.136	0.098
利率(非周期)	0.056	0.150	0.148	... ...	0.150	0.149	0.150	0.095
贷款额度	70 万	10 万	80 万	... ...	20 万	30 万	10 万	40 万

5.3 企业对突发因素的应对情况

在需要贷款的企业中，由于所处的行业不同，在面对不同类型的突发因素时，会受到不同程度的冲击，贷款企业的风险也会出现变化。因此，分析不同行业应对突发因素时的影响对于银行的信贷策略的调整有重要的意义。

5.3.1 新冠病毒疫情的影响

2019 年末，新冠疫情爆发，对中国的社会经济造成了很大的影响，最直接的影响就是对中小微企业的影响。

本文继续使用 5.1 节中的企业分类标准，仍将各个企业所涉及的行业分为：商业、技术行业、工业、商业、邮政物流业、生活服务业、文化传媒业、农林牧

渔业、个体户 9 个行业类别。各个行业受疫情冲击的影响分类如图 10 所示。由于互联网时代的便利性，技术行业所受到的冲击最小**[3]**。甚至在疫情期间，由于“禁足令”等政策的实施，许多实体经济的产品都在互联网上进行销售，促进了互联网行业企业的发展。与之相反的是工业、文化传媒行业、商业、生活服务业、个体户等需要线下交易或承担风险能力较弱的企业，它们在应对疫情时抗风险能力较弱。农林牧渔、邮政物流行业受到的冲击相对较小，对疫情得到影响并不十分敏感。

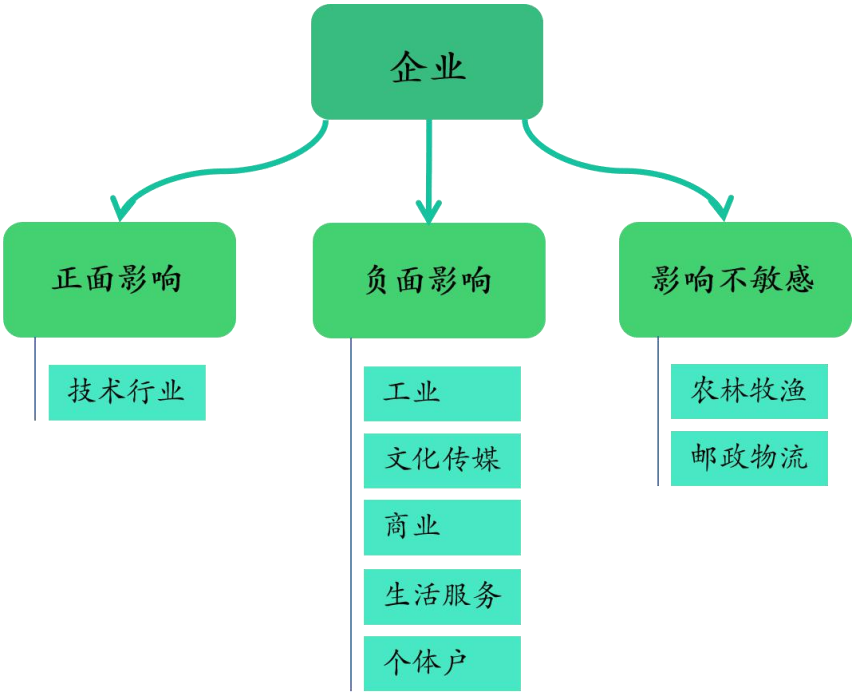


图 10 各个行业受疫情冲击的影响分类图

依据相关文献**[4]**，对原模型进行如下调整：

- **提高容忍度原则**

疫情期间，提高中小企业不良率容忍度对帮助中小微企业克服短期困难有一定的帮助。在疫情期间，中小微企业的现金流减少，进而会进一步对银行的贷款质量造成冲击，从而导致银行受影响行业的贷款动力。

为提高对企业的容忍度，银行在判断是否对贷款企业发放贷款时，将风险阈值r_0下调 5%。

- **专项额度原则**

对于银行用于贷款的总资产U，从中分出部分用于帮扶中小微企业，分配出用于帮扶的部分为：

$$U_{reserve} = \alpha U . \tag{40}$$

其中总资产U满足：

$$U = U_{reserve} + U_{allocate}. \tag{41}$$

将剩余的资产$U_{allocate}$使用 5.2 中的模型对企业进行分配，企业i从中分配到的贷款金额为$U_i^{(1)}$。为在疫情中帮扶有困难的中小微企业，本文对n个在疫情中受到产业冲击的企业分拨贷款资金$U_{reserve}$。

下面确定分拨贷款资金的分配策略。银行为企业分配贷款资金与行业受到的疫情冲击指数正相关，与该企业的风险等级负相关，定义银行为企业i分配贷款资金的权重：

$$q_i = \frac{k_j}{r_i}. \tag{42}$$

根据权重q_i得到银行分拨额外的贷款资金为：

$$U_i^{(2)} = \frac{q_i}{\sum q_i} U_{reserve}. \tag{43}$$

得到最终为贷款企业提供的贷款资金为：

$$U_i = U_i^{(1)} + U_i^{(2)}. \tag{44}$$

• 利率调整

受疫情冲击，银行需要对不同行业η_i的利率进行调整。经过调整后的企业利率为：

$$\eta_i = \begin{cases} \eta_i(1+\beta \cdot h_1(r_i))，\text{受正面影响企业} \\ \eta_i，\text{不敏感企业} \\ \eta_i(1-\alpha \cdot h_2(r_i))，\text{受负面影响企业} \end{cases}, \tag{45}$$

其中r_i表示企业的风险，$h_1(r_i)$、$h_2(r_i)$分别为正面冲击利率上调函数与负面冲击利率下调函数。二者的计算公式分别是：

$$h_1(r_i) = \frac{r_i - \min\{r_i\}}{\max\{r_i\} - \min\{r_i\}}, \tag{46}$$

$$h_2(r_i) = \frac{\max\{r_i\} - r_i}{\max\{r_i\} - \min\{r_i\}}. \tag{47}$$

参数α, β均为扰动因子，浮动的区间为 0.4~0.7。通过 Python 求解，得到企业贷款额度的变化对照表如表 5 所示。经统计，新的平均信贷利率相较之前下降了 3.639%。

表 5 企业贷款额度、利率的变化对照表

企业代号	E126	E127	E128	E129	E130	...	E421	E422	E423	E425
原始额度(万元)	70.00	30.00	50.00	40.00	10.00	...	10.00	10.00	50.00	50.00
原始利率	0.14	0.13	0.13	0.12	0.10	...	0.10	0.09	0.10	0.10
新利率	0.14	0.13	0.11	0.09	0.06	...	0.04	0.04	0.04	0.04
总额度	380.93	30.00	82.60	59.02	22.38	...	17.20	15.39	57.22	55.45

5.3.2 洪涝灾害的影响

自 2020 年入汛以来，我国的南方地区发生了多轮强降雨，造成多地发生较为严重的洪涝灾害。这次降雨对我国南方多地造成破坏，对社会经济发展造成一定的损失，不少企业在这次洪涝灾害中遭到重创或不同程度的冲击。

当洪涝灾害发生时，根据“附件 2”中所给出的企业类型，本文主要针对农林牧渔行业进行分析。如图 11 所示，洪涝灾害将直接导致农业企业减产，销量减少，同时也将对园林企业的行情造成影响。此外，农业企业也将对其下游企业(如餐饮行业、食品、农副产品企业)生产造成一定的影响。

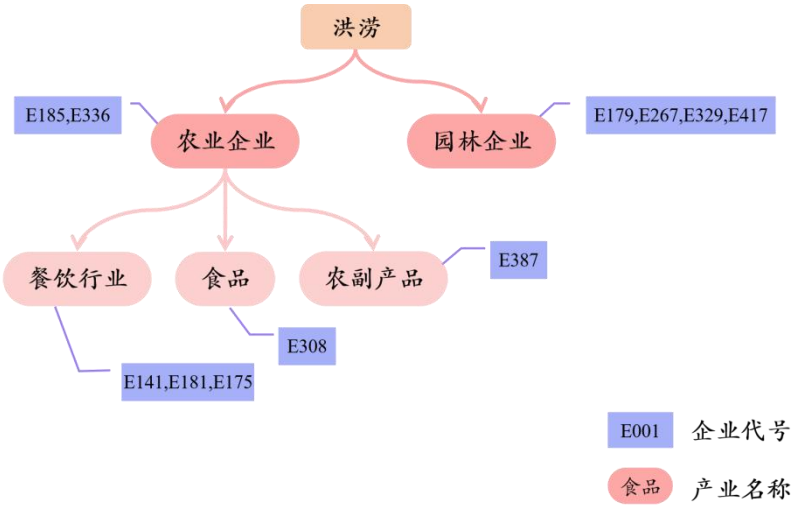


图 11 洪涝灾害影响分析图

本文定义洪涝灾害对企业的冲击系数 σ_i。对于洪涝灾害的直接影响的企业，本文认为 $\sigma_i = 1$。洪涝灾害的影响传播效应类似于食物链的能量传播关系，企业之间的影响符合逐级递减的规律。行业冲击在各个层级之间的传递效率为 e，本文中 $e = 30\%$。图 10 中的企业受到突发洪涝灾害影响后，向下一层级的企业传播的灾害影响为 $\sigma_i' = \sigma_i \cdot e$。

对于受到冲击的企业，对利率进行调整，经过调整后的利率为：

$$\eta_i' = \eta_i(1 - \sigma_i \cdot f(r_i)), \tag{48}$$

其中 $f(r_i)=\dfrac{\max\{r_i\}-r_i}{\max\{r_i\}-\min\{r_i\}}$.

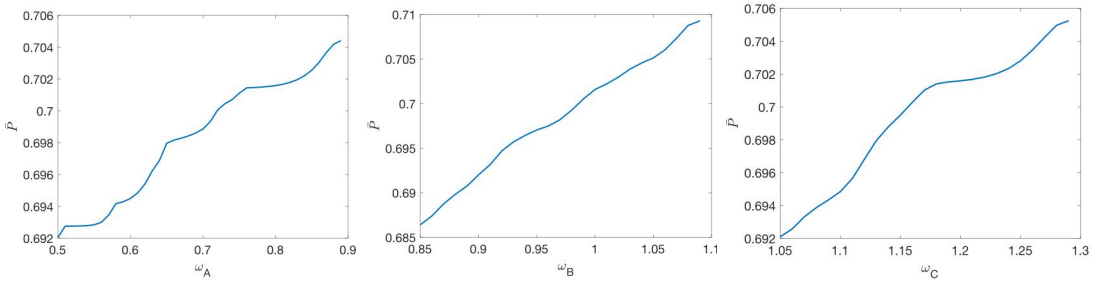
通过 Python 求解，得到企业贷款利率的变化对照表如表 6 所示。经统计，新的平均信贷利率相较之前下降了 0.160%。

表 6 洪涝灾害时利率表

企业代号	E126	E127	E128	E129	E130	...	E421	E422	E423	E425
原始额度(万元)	70.00	30.00	50.00	40.00	10.00	...	10.00	10.00	50.00	50.00
洪涝灾害利率	0.14	0.13	0.13	0.12	0.10	...	0.10	0.09	0.10	0.10

5.4 灵敏度分析

对本文的优化模型进行灵敏度分析。在银行的贷款策略模型中，本文建立了企业还款概率矩阵 $P_{m\times 10}$ 对企业的还款概率 $P_{i,j}$ 进行刻画。企业的还款概率 $P_{i,j}$ 涉及到参数的 $\omega_A,\omega_B,\omega_C$，均会对公式（24）的计算产生影响。下面分别研究 ω_A, ω_B, ω_C 对企业平均还款概率 $\overline{P}$ 的影响。其中，企业平均还款概率 $\overline{P}$ 为企业还款概率矩阵所有元素的均值。平均还款概率反映的是所研究的企业在贷款一定金额时还款的可能性，该变量对单目标优化模型有重要的影响。



(a) (b) (c)

图 12 企业平均还款概率与 ω_A, ω_B, ω_C 的关系

当参数 ω_A 变化范围为 0.5~0.9 之间时，企业平均还款概率 $\overline{P}$ 的变化范围不超过 0.014；当参数 ω_B 变化范围为 0.85~1.1 之间时，企业平均还款概率 $\overline{P}$ 的变化范围不超过 0.025；当参数 ω_C 变化范围为 1.05~1.3 之间时，企业平均还款概率 $\overline{P}$ 的变化范围不超过 0.014。

因此，在参数 ω_A, ω_B, ω_C 浮动时，所有企业的平均还款概率的波动较小。可以体现参数对还款概率的影响并不敏感，模型对于参数的变化是稳定的。

六、模型的评价

6.1 模型的优点

1、第一问模型中，遵循系统性、针对性原则，充分挖掘数据中的信息，共确定了 10 个反映信贷风险的指标，计算得到的信贷风险与实际的履约概率具有良好的相关性，说明我们的量化方式具有优秀的刻画能力。

2、第二问模型中，在承接第一问模型的基础上，我们考虑对年累计利润变化具有周期性的企业颁布优惠政策，进一步优化模型，另辟蹊径，通过增大这些企业的“黏性”，提高银行的总利润，想法充分体现了创新性。

3、第三问模型中，在刻画新冠病毒对企业的影响时，根据“提升银行对不良企业容忍度原则”、“专项额度原则”和“利率调整原则”，对银行的信贷策略做出调整；在反映旱涝灾害对企业的影响时，我们将“上下游产业链”类比“生物链”，成功刻画出了其对不同级别企业的影响，调整方案具有考虑面广、创新性强的特点。

6.2 模型的缺点

由于专业数据的缺乏，部分参数的确定具有主观性，可能会对结果的精确度产生影响。

七、模型的改进与推广

7.1 模型的改进

1、在专业性数据充足的条件下，可用更为专业的数据作为模型中的参数，从而对模型结果做出修正。

2、在考虑银行的信贷策略时，可以引入动态的信贷策略调整模型，根据实时的企业信贷风险评估，确定更为精确的信贷策略。

7.2 模型的推广

1、本文所述“企业信贷风险评估模型”，不仅可用来评估中小企业的信贷风险，还可以推广到更广泛的风险评定问题。

2、本文构造的“银行信贷针对突发因素的信贷策略调整模型”主要针对的是新冠病毒和旱涝灾害两大风险，通过修改模型中的相关参数，也可以推广用于确定其他突发风险下的信贷调整策略。

八、参考文献

[1]: 蒙震，中小企业信用风险评估及其对贷款定价影响研究[D].对外经济贸易大学:对外经济贸易大学，2013.

[2]: 程序员大本营，王芳，时间序列如何更好地分析周期性 https://www.pianshen.com/article/51981538217/，2020.09.13.

[3]:何诚颖，闻岳春，常雅丽，耿晓旭，新冠病毒肺炎疫情对中国经济影响的测度分析[J]，数量经济技术经济研究，第五期：3-21，2020.

[4]:新冠肺炎疫情对中小企业的冲击与金融纾困，清华金融评论，

https://www.weiyangx.com/366061.html,2020.09.13.

[5]马九杰,吴本健.利率浮动政策、差别定价策略与金融机构对农户的信贷配给[J].金融研究,2012(04):155-168.

附录

附录 A 支撑材料文件列表

说明：因为题目所给的完整数据集过大无法放入支撑材料，所以在运行代码前烦请将下面使用***加粗斜体***标注的四个文件放入相应位置。另外 python 代码在运行时请设定工作目录在支撑材料根目录，MATLAB 代码在运行时请设定工作目录在相应代码所在目录。

.

├── ***data1.xlsx　对应题目附件 1（因为完整数据集过大，并没有压缩包内）***
├── data1_info.xlsx　题目附件 1 的企业信息的 sheet（在原基础上有修改）
├── ***data2.xlsx　题目附件 2（因为完整数据集过大，并没有压缩包内）***
├── data2_info.xlsx　题目附件 2 的企业信息的 sheet（在原基础上有修改）
├── ***data3.xlsx　题目附件 3（因为完整数据集过大，并没有压缩包内）***
├── problem1
│　　├── part1　问题一各项指标计算
│　　│　　├── predeal.py　各项指标计算
│　　│　　└── shang.py　熵权法降维
│　　├── part2　问题一最终结果计算代码
│　　│　　├── fit.m　利率和损失率拟合代码
│　　│　　├── fun.m　利率和损失率拟合代码
│　　│　　├── income_old.m　单目标优化目标函数
│　　│　　├── predeal.py　概率矩阵生成
│　　│　　├── solve.m　单目标优化求解
│　　│　　└── 拟合.svg
│　　├── problem1_result.csv　第一问求解结果
│　　└── result　问题一中间结果
│　　　　├── average_income_year.xlsx　平均每年净利润
│　　　　├── lambda.xlsx　SPSS 因子分析得到的系数
│　　　　├── problem1_part1.xlsx　问题一所用的各项指标
│　　　　└── risk_rate.csv　风险系数
├── problem2
│　　├── part1　问题二各项指标计算
│　　│　　├── label_predit.py　指标恢复代码
│　　│　　├── predeal1_concat.py　拼接信誉度和其他指标
│　　│　　├── predeal1_getres.py　决策树预测输入数据集生成代码
│　　│　　├── predeal1_train.py　决策树训练集生成代码
│　　│　　├── predeal2.py　风险系数计算
│　　│　　└── shang.py　熵权法降维
│　　├── part2
│　　│　　├── income_old.m　单目标优化目标函数
│　　│　　└── solve.m　单目标优化求解

```
│   ├── ploblem2_result.csv  问题二求解完整结果
│   ├── predict_result.csv  指标恢复结果
│   └── result  问题二中间结果
│       ├── lambda.xlsx  SPSS 因子分析得到的系数
│       ├── predit_label.csv  指标恢复初步结果（不含化公司代号）
│       ├── problem2_part1_getres.xlsx  生成的决策树预测输入
│       ├── problem2_part1_res.xlsx  问题二所用各项指标计算结果
│       ├── problem2_part1_train.xlsx  生成的决策树训练集
│       ├── risk_rate.csv  风险系数
│       └── risk_rate_add.csv  包含公司代号的风险系数
├── problem3
│   ├── problem3_result.xlsx  问题三最终结果
│   ├── risk_rate.xlsx  原始额度和原始利率结果
│   └── solve.py  问题三求解
└── sensitivity
    ├── paint.m  灵敏度分析做图（基于代码的结果）
    └── analyze.py  灵敏度分析代码
```

附录 B 完整数据

1、第一问

代号	利率	额度
E1	0.099675	10
E10	0.063234	100
E104	0.06529	20
E105	0.066856	10
E106	0.060771	90
E11	0.063016	100
E110	0.066256	10
E12	0.15	10
E13	0.087834	10
E14	0.061125	100
E15	0.081231	10
E16	0.13142	10
E17	0.13162	10
E18	0.12423	10
E19	0.10757	10
E2	0.14985	10
E20	0.068586	100
E21	0.062228	100
E22	0.098698	10
E23	0.063472	100
E24	0.090725	10
E25	0.061028	90
E26	0.076244	20
E27	0.08315	10
E28	0.061283	100
E3	0.071333	100
E30	0.061457	100
E31	0.068984	40
E32	0.061169	100
E33	0.062254	100
E34	0.063108	100
E35	0.061233	100
E37	0.061504	100
E38	0.061472	100
E39	0.060768	100
E4	0.064034	10
E40	0.062237	50
E41	0.062417	50
E42	0.07508	20
E43	0.061451	100
E44	0.062341	50
E46	0.062744	40

E47	0.063287	40
E48	0.071274	30
E49	0.061235	80
E5	0.067454	100
E50	0.06267	50
E51	0.061341	100
E53	0.062248	50
E54	0.071132	30
E55	0.06132	80
E56	0.062523	50
E57	0.063938	100
E58	0.060965	100
E59	0.070383	30
E6	0.13968	10
E60	0.060847	100
E61	0.060778	100
E62	0.060851	100
E63	0.061409	100
E64	0.066245	60
E65	0.06083	100
E66	0.061016	100
E67	0.060786	100
E68	0.062683	50
E69	0.065873	20
E7	0.10778	10
E70	0.060805	100
E71	0.061212	100
E72	0.064607	20
E73	0.06508	20
E74	0.060927	100
E75	0.067057	10
E76	0.060782	100
E77	0.064353	30
E78	0.065116	20
E79	0.060832	100
E8	0.10915	10
E80	0.063987	30
E81	0.06755	50
E83	0.061256	100
E84	0.067921	50
E85	0.060773	90
E86	0.061628	70
E88	0.067507	50
E89	0.069178	40
E9	0.08049	20
E90	0.066996	10
E91	0.067617	50
E92	0.065646	20

E93	0.060767	90
E94	0.067274	10
E95	0.060895	100
E96	0.061285	100
E97	0.060756	90
E98	0.060755	90

2、第二问缺失指标恢复

编号	信誉评级	是否违约
E124	A	否
E125	A	否
E126	A	否
E127	C	否
E128	A	否
E129	A	否
E130	A	否
E131	A	否
E132	A	否
E133	A	否
E134	A	否
E135	A	否
E136	A	否
E137	A	否
E138	A	否
E139	A	否
E140	A	否
E141	A	否
E142	A	否
E143	A	否
E144	A	否
E145	A	否
E146	A	否
E147	A	否
E148	A	否
E149	A	否
E150	A	否
E151	A	否
E152	A	否
E153	A	否
E154	A	否
E155	A	否
E156	A	否

E157	A	否
E158	A	否
E159	A	否
E160	A	否
E161	A	否
E162	A	否
E163	A	否
E164	A	否
E165	A	否
E166	A	否
E167	A	否
E168	A	否
E169	A	否
E170	A	否
E171	A	否
E172	A	否
E173	A	否
E174	A	否
E175	A	否
E176	A	否
E177	A	否
E178	A	否
E179	A	否
E180	A	否
E181	A	否
E182	A	否
E183	A	否
E184	A	否
E185	A	否
E186	A	否
E187	A	否
E188	A	否
E189	A	否
E190	A	否
E191	A	否
E192	A	否
E193	A	否
E194	A	否
E195	A	否
E196	A	否
E197	A	否
E198	A	否

E199	A	否
E200	A	否
E201	A	否
E202	A	否
E203	A	否
E204	A	否
E205	A	否
E206	A	否
E207	A	否
E208	A	否
E209	A	否
E210	A	否
E211	A	否
E212	A	否
E213	A	否
E214	A	否
E215	A	否
E216	A	否
E217	A	否
E218	A	否
E219	A	否
E220	A	否
E221	A	否
E222	A	否
E223	A	否
E224	A	否
E225	A	否
E226	A	否
E227	A	否
E228	A	否
E229	A	否
E230	A	否
E231	A	否
E232	A	否
E233	C	否
E234	A	否
E235	A	否
E236	A	否
E237	A	否
E238	A	否
E239	A	否
E240	A	否

E241	A	否
E242	C	否
E243	A	否
E244	A	否
E245	C	否
E246	A	否
E247	A	否
E248	A	否
E249	A	否
E250	A	否
E251	C	否
E252	A	否
E253	A	否
E254	A	否
E255	A	否
E256	A	否
E257	A	否
E258	A	否
E259	A	否
E260	A	否
E261	C	否
E262	A	否
E263	A	否
E264	A	否
E265	A	否
E266	C	否
E267	A	否
E268	C	否
E269	A	否
E270	C	否
E271	A	否
E272	C	否
E273	A	否
E274	C	否
E275	C	否
E276	C	否
E277	A	否
E278	C	否
E279	A	否
E280	C	否
E281	A	否
E282	A	否

E283	A	否
E284	C	否
E285	A	否
E286	A	否
E287	B	否
E288	A	否
E289	C	否
E290	C	否
E291	C	否
E292	C	否
E293	C	否
E294	B	否
E295	C	否
E296	C	否
E297	C	否
E298	C	否
E299	A	否
E300	A	否
E301	C	否
E302	B	否
E303	C	否
E304	C	否
E305	A	否
E306	C	否
E307	C	否
E308	C	否
E309	C	否
E310	C	否
E311	C	否
E312	A	否
E313	C	否
E314	C	否
E315	C	否
E316	C	否
E317	C	否
E318	C	否
E319	C	否
E320	C	否
E321	C	否
E322	C	否
E323	B	否
E324	A	否

E325	A	否
E326	C	否
E327	C	否
E328	C	否
E329	C	否
E330	C	否
E331	C	否
E332	A	否
E333	B	否
E334	C	否
E335	C	否
E336	C	否
E337	C	否
E338	C	否
E339	C	否
E340	C	否
E341	C	否
E342	C	否
E343	C	否
E344	C	否
E345	C	否
E346	C	否
E347	C	否
E348	C	否
E349	C	否
E350	C	否
E351	C	否
E352	B	否
E353	C	否
E354	C	否
E355	A	否
E356	C	否
E357	C	否
E358	C	否
E359	C	否
E360	C	否
E361	C	否
E362	C	否
E363	B	否
E364	C	否
E365	C	否
E366	C	否

E367	C	否
E368	C	否
E369	C	否
E370	C	否
E371	C	否
E372	C	否
E373	C	否
E374	A	否
E375	A	否
E376	C	否
E377	A	否
E378	C	否
E379	B	否
E380	C	否
E381	A	否
E382	A	是
E383	C	否
E384	B	否
E385	D	是
E386	C	否
E387	C	否
E388	A	否
E389	C	否
E390	A	否
E391	A	否
E392	A	否
E393	A	否
E394	B	否
E395	C	否
E396	C	否
E397	A	否
E398	C	是
E399	C	是
E400	C	是
E401	D	是
E402	C	否
E403	C	否
E404	C	否
E405	C	是
E406	C	否
E407	C	否
E408	C	是

E409	D	是
E410	D	是
E411	D	是
E412	D	是
E413	D	是
E414	D	是
E415	D	是
E416	D	是
E417	D	是
E418	D	是
E419	B	否
E420	A	否
E421	D	是
E422	D	是
E423	D	是
E424	D	是
E425	D	是

3、第二问最终结果

代号	周期		非周期	
E124	0.089913	70	0.056342	70
E125	0.13532	10	0.14985	10
E126	0.13547	10	0.14842	80
E127	0.13549	30	0.1436	20
E128	0.13541	30	0.14995	10
E129	0.13538	20	0.14838	70
E130	0.13529	10	0.14893	70
E131	0.13542	30	0.149	40
E132	0.13545	50	0.14481	30
E133	0.13538	20	0.14996	10
E134	0.13529	10	0.14761	20
E135	0.13538	20	0.149	40
E136	0.13545	50	0.1472	80
E137	0.1355	20	0.14906	30
E138	0.1353	10	0.14891	60
E139	0.13547	70	0.14891	70
E140	0.1355	80	0.14893	70
E141	0.13546	60	0.14783	30
E142	0.1355	20	0.14834	60
E143	0.13545	50	0.14996	10
E144	0.13547	90	0.14923	20
E145	0.1355	20	0.14996	10

E146	0.13549	10	0.14897	50
E147	0.1355	70	0.14923	20
E148	0.1355	90	0.14996	10
E149	0.13542	30	0.14838	70
E150	0.13549	10	0.14892	50
E151	0.13529	10	0.14923	20
E152	0.13547	80	0.14893	80
E153	0.13543	40	0.14799	40
E154	0.1353	10	0.14906	30
E155	0.10749	20	0.14996	10
E156	0.13546	60	0.14996	10
E157	0.13547	70	0.14996	10
E158	0.13544	40	0.14893	70
E159	0.13547	80	0.14751	20
E160	0.13545	50	0.14903	20
E161	0.13549	20	0.14892	50
E162	0.13546	70	0.14838	70
E163	0.13529	10	0.14906	30
E164	0.13547	80	0.14892	50
E165	0.13547	70	0.14828	70
E166	0.13547	80	0.14897	50
E167	0.1355	20	0.1489	80
E168	0.1355	40	0.14994	10
E169	0.13546	60	0.14897	50
E170	0.13547	90	0.14624	50
E171	0.1353	10	0.14994	10
E172	0.13538	20	0.14887	10
E173	0.13529	10	0.14887	10
E174	0.13547	80	0.14996	10
E175	0.13549	10	0.14891	70
E176	0.1353	10	0.14838	70
E177	0.1355	60	0.14857	20
E178	0.13544	40	0.14996	10
E179	0.13542	30	0.14923	20
E180	0.13547	70	0.14996	10
E181	0.1353	10	0.14996	10
E182	0.13529	10	0.14892	50
E183	0.12719	90	0.14906	30
E184	0.13545	50	0.149	40
E185	0.13547	90	0.14751	20
E186	0.13529	10	0.14895	50
E187	0.12719	90	0.14696	70

E188	0.1355	60	0.14994	10
E189	0.13542	30	0.14903	20
E190	0.13548	10	0.14903	20
E191	0.13538	20	0.149	40
E192	0.13529	10	0.14994	10
E193	0.13529	10	0.14906	30
E194	0.13545	50	0.14996	10
E195	0.1355	90	0.14891	70
E196	0.13546	60	0.14906	30
E197	0.13529	10	0.14895	60
E198	0.13547	80	0.14901	20
E199	0.13529	10	0.14996	10
E200	0.13547	70	0.14666	60
E201	0.13538	20	0.14891	70
E202	0.13547	80	0.14996	10
E203	0.13544	40	0.14827	30
E204	0.13544	40	0.14996	10
E205	0.1188	40	0.14887	10
E206	0.1355	30	0.14996	10
E207	0.13542	30	0.1489	80
E208	0.13542	30	0.14906	30
E209	0.13545	50	0.14995	20
E210	0.1353	10	0.14895	60
E211	0.13542	30	0.14761	20
E212	0.13547	80	0.14799	40
E213	0.13544	40	0.14666	60
E214	0.13547	80	0.14892	50
E215	0.1353	10	0.14893	80
E216	0.13538	20	0.14834	60
E217	0.1355	80	0.14994	10
E218	0.13547	70	0.14897	30
E219	0.13547	80	0.14923	20
E220	0.1353	10	0.14838	70
E221	0.13542	30	0.14829	50
E222	0.13546	60	0.14812	50
E223	0.14998	90	0.14887	10
E224	0.1355	40	0.14894	40
E225	0.13545	50	0.14361	20
E226	0.13546	60	0.14906	20
E227	0.13547	70	0.14996	10
E228	0.13547	80	0.14893	70
E229	0.1355	40	0.14893	80

E230	0.13546	60	0.14891	70
E231	0.13547	70	0.14894	40
E232	0.13544	40	0.14906	30
E233	0.13545	50	0.14996	10
E234	0.14858	60	0.14996	10
E235	0.1353	10	0.149	40
E236	0.13538	20	0.14783	30
E237	0.13544	40	0.14996	10
E238	0.13529	10	0.14827	30
E239	0.1188	40	0.14996	10
E240	0.13547	70	0.14666	60
E241	0.13546	60	0.14907	30
E242	0.13538	20	0.14842	80
E243	0.1353	10	0.14566	40
E244	0.13545	50	0.14696	70
E245	0.13546	60	0.14566	40
E246	0.13542	30	0.14482	30
E247	0.13547	80	0.14894	40
E248	0.13544	40	0.14893	80
E249	0.13544	40	0.14996	10
E250	0.13547	80	0.14751	20
E251	0.13538	20	0.14996	10
E252	0.13538	20	0.14624	50
E253	0.13547	80	0.14897	50
E254	0.1355	80	0.14996	10
E255	0.13547	70	0.14624	50
E256	0.13545	50	0.14888	10
E257	0.13547	90	0.14897	50
E258	0.10749	20	0.14994	10
E259	0.1355	70	0.14923	20
E260	0.13538	20	0.14923	20
E261	0.1353	10	0.14893	70
E262	0.13538	20	0.14996	10
E263	0.13547	90	0.14892	60
E264	0.13542	30	0.14566	40
E265	0.13547	90	0.14799	40
E266	0.13538	20	0.14829	50
E267	0.12361	60	0.14994	10
E268	0.13546	60	0.14827	30
E269	0.13546	60	0.14857	20
E270	0.1355	70	0.14887	10
E271	0.13549	10	0.14751	20

E272	0.13549	10	0.14895	60
E273	0.13545	50	0.14894	40
E274	0.13544	40	0.14923	20
E275	0.1355	60	0.14857	20
E276	0.13547	80	0.14482	30
E277	0.13542	30	0.14361	20
E278	0.1355	30	0.14895	60
E279	0.13547	70	0.14897	30
E280	0.13538	20	0.14996	10
E281	0.13545	50	0.14834	80
E282	0.1355	80	0.14996	10
E283	0.1353	10	0.14903	20
E284	0.13545	50	0.14895	60
E285	0.13547	70	0.14892	50
E286	0.1353	10	0.14996	10
E287	0.13538	20	0.149	40
E288	0.1353	10	0.14903	20
E289	0.13538	20	0.14923	20
E290	0.13538	20	0.14783	30
E291	0.13544	40	0.1489	80
E292	0.1355	60	0.14842	80
E293	0.13542	30	0.14891	60
E294	0.13544	40	0.1472	80
E295	0.13549	10	0.14923	20
E296	0.13546	60	0.14696	70
E297	0.13542	30	0.14361	20
E298	0.13547	80	0.14994	10
E299	0.13547	90	0.14995	20
E300	0.1355	20	0.14566	40
E301	0.13547	70	0.14482	30
E302	0.13547	90	0.14895	60
E303	0.13546	60	0.14892	60
E304	0.13545	50	0.14892	60
E305	0.13544	40	0.14857	20
E306	0.13549	10	0.14888	10
E307	0.13544	40	0.14994	10
E308	0.13546	60	0.14361	20
E309	0.13542	30	0.14897	30
E310	0.11458	30	0.14897	30
E311	0.1353	10	0.14906	30
E312	0.1353	10	0.14996	10
E313	0.1353	10	0.14834	60

E314	0.13549	10	0.14897	30
E315	0.1355	50	0.14893	70
E316	0.13547	90	0.14825	40
E317	0.13547	90	0.14996	10
E318	0.13547	80	0.14812	50
E319	0.14859	70	0.14751	20
E320	0.13546	70	0.14838	70
E321	0.1355	80	0.14751	20
E322	0.13538	20	0.14897	50
E323	0.13915	20	0.149	40
E324	0.13547	90	0.14892	50
E325	0.1355	30	0.14783	30
E326	0.1355	80	0.14996	10
E327	0.1355	80	0.14906	30
E328	0.1355	40	0.14751	20
E329	0.1355	90	0.14923	20
E330	0.1353	10	0.14892	50
E331	0.1353	10	0.14895	50
E332	0.1355	80	0.14799	40
E333	0.1355	30	0.14996	10
E334	0.13549	10	0.14996	10
E335	0.13545	50	0.14888	10
E336	0.1353	10	0.14892	60
E337	0.12361	60	0.14892	60
E338	0.13542	30	0.14624	50
E339	0.13542	30	0.14907	30
E340	0.13546	60	0.14906	20
E341	0.13547	80	0.14888	10
E342	0.1355	60	0.14996	10
E343	0.1355	30	0.14996	10
E344	0.13546	60	0.14891	70
E345	0.1355	10	0.14666	60
E346	0.13538	20	0.14893	70
E347	0.1353	10	0.1489	80
E348	0.1353	10	0.14895	60
E349	0.13547	80	0.14996	10
E350	0.1353	10	0.14996	10
E351	0.1353	10	0.14892	60
E352	0.1251	70	0.14624	50
E353	0.12626	80	0.14891	70
E354	0.13538	20	0.14893	70
E355	0.1355	40	0.14566	40

E356	0.13547	70	0.14834	60
E357	0.1355	90	0.14838	70
E358	0.1355	40	0.14903	20
E359	0.1353	10	0.14968	10
E360	0.13547	70	0.14907	30
E361	0.1355	20	0.14996	10
E362	0.1353	10	0.14888	10
E363	0.1355	70	0.14996	10
E364	0.1355	30	0.14897	30
E365	0.1353	10	0.14761	20
E366	0.1355	20	0.14994	10
E367	0.1353	10	0.14892	60
E368	0.13547	70	0.14996	10
E369	0.1355	40	0.14897	30
E370	0.1353	10	0.14854	20
E371	0.13547	90	0.14923	20
E372	0.13538	20	0.14829	50
E373	0.1353	10	0.14827	30
E374	0.13545	50	0.14897	50
E375	0.13547	70	0.14761	20
E376	0.13847	60	0.14751	20
E377	0.1355	10	0.14888	10
E378	0.13546	60	0.149	40
E379	0.13549	10	0.149	40
E380	0.13546	60	0.14825	40
E381	0.13547	70	0.14482	30
E383	0.13547	90	0.14897	50
E384	0.1355	70	0.14761	20
E386	0.13547	80	0.14996	10
E387	0.13545	50	0.14996	10
E388	0.13547	80	0.14891	70
E389	0.1353	10	0.14996	10
E390	0.1353	10	0.14959	20
E391	0.13546	60	0.14996	10
E392	0.13549	10	0.14996	10
E393	0.1355	60	0.14842	80
E394	0.13547	90	0.149	40
E395	0.13538	20	0.14838	70
E396	0.13545	50	0.14894	40
E397	0.13538	20	0.14834	60
E402	0.1355	30	0.14996	10
E403	0.1355	50	0.14907	30

E404	0.13547	80	0.1472	80
E406	0.1355	20	0.14996	10
E407	0.13542	30	0.14888	10
E419	0.1355	10	0.14996	10
E420	0.097966	40	0.095211	10

4、

企业代号	原始额度	原始利率	新冠额度	新冠利率	洪灾利率
E126	70	0.13836	380.9271	0.136202	0.13836
E127	30	0.1275	30	0.1275	0.1275
E128	50	0.13093	82.5972	0.111448	0.13093
E129	40	0.12322	59.02142	0.091799	0.12322
E130	10	0.10231	22.37995	0.062225	0.10231
E131	60	0.13414	84.57175	0.107661	0.13414
E132	60	0.12226	72.67101	0.075459	0.12226
E133	10	0.10315	23.27439	0.065459	0.10315
E134	30	0.10711	36.83934	0.056468	0.10711
E135	10	0.098461	20.28022	0.057166	0.098461
E137	50	0.12149	62.60679	0.07994	0.12149
E138	20	0.10307	30.623	0.056008	0.10307
E139	50	0.11661	61.63181	0.067984	0.11661
E140	30	0.11237	42.26258	0.072861	0.11237
E141	30	0.10895	40.17829	0.062799	0.10895
E142	10	0.10026	10	0.10026	0.10026
E143	10	0.099383	10	0.115306	0.099383
E144	10	0.098327	17.24382	0.054434	0.098327
E147	60	0.12557	72.90102	0.083605	0.12557
E148	70	0.11961	77.32256	0.06679	0.11961
E149	40	0.10745	49.08454	0.056454	0.10745
E150	60	0.11816	69.98038	0.067115	0.11816
E151	70	0.12491	81.2808	0.077169	0.12491
E152	40	0.11265	40	0.11265	0.11265
E153	50	0.13104	83.82471	0.112249	0.13104
E156	20	0.10861	35.63657	0.074919	0.10861
E157	20	0.10163	29.03883	0.053152	0.10163
E158	10	0.10105	20.16389	0.058185	0.10105
E159	50	0.11386	60.85115	0.062965	0.11386
E160	10	0.10257	23.79246	0.070507	0.10257
E161	60	0.12607	70.2072	0.086131	0.12607
E163	80	0.12689	80	0.12689	0.12689
E164	20	0.10522	31.66537	0.06147	0.10522
E165	10	0.10078	19.79383	0.06196	0.10078
E166	10	0.10076	19.92524	0.05699	0.10076
E168	60	0.12105	70.35428	0.070645	0.12105
E169	40	0.11158	49.84366	0.062708	0.11158
E170	20	0.10742	30.30971	0.062497	0.10742
E171	70	0.11833	70	0.143479	0.11833

E172	50	0.10833	50	0.10833	0.10833
E174	40	0.10775	49.01615	0.056224	0.10775
E176	30	0.10378	30	0.13432	0.10378
E177	60	0.11524	69.40559	0.062414	0.11524
E179	10	0.098281	10	0.098281	0.052011
E180	20	0.10241	29.34444	0.055158	0.10241
E183	10	0.10069	19.23634	0.053688	0.10069
E184	10	0.09807	18.4739	0.048172	0.09807
E185	20	0.097486	20	0.097486	0.042545
E186	60	0.11703	69.72378	0.065139	0.11703
E189	10	0.097791	18.33628	0.047213	0.097791
E190	70	0.12715	79.18127	0.082368	0.12715
E192	40	0.11789	52.22087	0.076298	0.11789
E193	10	0.10542	26.53191	0.07449	0.10542
E194	10	0.10249	19.80911	0.063072	0.10249
E195	60	0.12072	70.7092	0.072118	0.12072
E196	20	0.10082	28.75258	0.051156	0.10082
E197	60	0.12283	70.80582	0.073821	0.12283
E199	10	0.099263	19.02748	0.051855	0.099263
E201	50	0.1078	50	0.1078	0.1078
E202	10	0.097959	21.75015	0.057522	0.097959
E203	80	0.11194	88.26913	0.053574	0.11194
E204	70	0.12043	70	0.137756	0.12043
E205	30	0.10404	39.94355	0.053289	0.10404
E206	60	0.11295	68.98234	0.058734	0.11295
E207	40	0.11562	51.76726	0.067961	0.11562
E209	50	0.10919	58.80621	0.055731	0.10919
E210	30	0.10452	38.94878	0.054162	0.10452
E211	10	0.097953	19.42203	0.047527	0.097953
E212	60	0.10747	68.27865	0.0515	0.10747
E213	30	0.10428	30	0.133092	0.10428
E214	80	0.1166	88.74194	0.059093	0.1166
E215	20	0.097655	28.34596	0.047206	0.097655
E216	10	0.098303	18.54444	0.048699	0.098303
E220	50	0.11337	58.48534	0.062966	0.11337
E221	10	0.098756	16.84469	0.048213	0.098756
E222	10	0.10115	10	0.10115	0.10115
E225	60	0.11859	68.20017	0.067928	0.11859
E227	40	0.10503	40	0.142806	0.10503
E228	70	0.11326	77.6866	0.057672	0.11326
E230	20	0.097898	27.95778	0.044857	0.097898
E232	50	0.12129	58.50064	0.075151	0.12129
E233	50	0.10294	55.96833	0.047167	0.10294
E234	10	0.096346	15.90943	0.043625	0.096346
E235	60	0.10537	69.05275	0.048913	0.10537
E236	10	0.096448	19.18648	0.045523	0.096448
E237	20	0.1041	33.63255	0.067061	0.1041
E238	20	0.10755	33.56393	0.06909	0.10755

E241	10	0.095752	18.97237	0.043988	0.095752
E243	10	0.096641	10	0.143566	0.096641
E244	30	0.10245	39.19693	0.048418	0.10245
E245	40	0.099996	47.78534	0.044618	0.099996
E246	40	0.10148	46.48724	0.046681	0.10148
E247	40	0.10086	47.89656	0.04579	0.10086
E248	60	0.10283	60	0.152256	0.10283
E249	10	0.10047	19.80337	0.056283	0.10047
E250	40	0.10204	48.07401	0.047551	0.10204
E251	30	0.099642	37.94019	0.045536	0.099642
E252	40	0.10333	48.24068	0.049268	0.10333
E253	60	0.10259	67.78534	0.045776	0.10259
E254	10	0.096	17.94897	0.04393	0.096
E255	10	0.097768	19.33499	0.046968	0.097768
E256	10	0.0967	17.7434	0.042857	0.0967
E257	50	0.10322	57.99318	0.047543	0.10322
E258	20	0.10107	20	0.10107	0.10107
E259	20	0.10005	27.14504	0.047224	0.10005
E260	10	0.096846	16.92471	0.044084	0.096846
E261	10	0.095751	17.76851	0.042609	0.095751
E262	40	0.10021	48.77753	0.044834	0.10021
E263	50	0.10528	50	0.149859	0.10528
E265	40	0.10238	48.08309	0.047771	0.10238
E266	30	0.09923	37.87061	0.044872	0.09923
E267	80	0.10571	80	0.10571	0.030151
E268	20	0.09876	28.11963	0.046318	0.09876
E269	20	0.1013	20	0.132435	0.1013
E270	10	0.095914	18.75851	0.042797	0.095914
E271	40	0.1039	40	0.143818	0.1039
E272	20	0.097434	28.81582	0.043826	0.097434
E273	10	0.096018	18.85444	0.043419	0.096018
E274	30	0.097551	37.57205	0.042005	0.097551
E275	50	0.10169	50	0.150401	0.10169
E276	70	0.10019	77.52447	0.042781	0.10019
E277	50	0.10171	57.84484	0.04581	0.10171
E278	30	0.097968	35.72731	0.042655	0.097968
E279	20	0.098734	28.20305	0.046839	0.098734
E280	20	0.096214	28.54557	0.041603	0.096214
E282	10	0.098402	16.43127	0.048925	0.098402
E283	40	0.099316	47.68542	0.0436	0.099316
E284	70	0.10131	77.6041	0.043867	0.10131
E285	40	0.10266	49.01238	0.047408	0.10266
E288	60	0.10176	68.67393	0.044856	0.10176
E289	10	0.096455	10	0.096455	0.096455
E290	40	0.099377	47.69365	0.043686	0.099377
E292	10	0.095948	10	0.095948	0.095948
E294	10	0.096433	18.6461	0.042334	0.096433
E295	50	0.10043	57.63642	0.043727	0.10043

E296	20	0.10005	28.15649	0.047164	0.10005
E297	10	0.098074	18.41437	0.047821	0.098074
E298	70	0.10573	76.40891	0.047938	0.10573
E300	20	0.097156	27.73506	0.043001	0.097156
E301	30	0.097502	35.66114	0.041809	0.097502
E302	10	0.095478	17.56407	0.041056	0.095478
E303	40	0.098302	46.13935	0.042211	0.098302
E305	10	0.098332	16.77912	0.051428	0.098332
E306	20	0.096568	28.50064	0.041466	0.096568
E307	20	0.096092	20	0.149034	0.096092
E308	10	0.097197	16.37393	0.043778	0.076154
E309	10	0.095847	18.73957	0.042652	0.095847
E310	50	0.10027	58.63686	0.043958	0.10027
E311	10	0.096034	19.08326	0.044752	0.096034
E313	70	0.10247	70	0.10247	0.10247
E314	60	0.10713	66.09661	0.050308	0.10713
E315	10	0.095322	17.41576	0.039902	0.095322
E316	10	0.096007	18.51855	0.041341	0.096007
E317	10	0.095459	10	0.147584	0.095459
E318	10	0.095316	15.60812	0.040357	0.095316
E319	10	0.095406	18.4828	0.040853	0.095406
E320	20	0.10382	29.15398	0.054921	0.10382
E321	60	0.10056	60	0.152347	0.10056
E323	40	0.098243	47.58003	0.042362	0.098243
E326	10	0.096169	17.62022	0.041757	0.096169
E327	50	0.102	58.83509	0.046002	0.102
E328	70	0.10093	70	0.154063	0.10093
E329	10	0.095566	10	0.095566	0.010837
E330	60	0.098802	65.60229	0.041773	0.098802
E331	10	0.095316	16.55645	0.040471	0.095316
E332	20	0.099432	26.34047	0.048722	0.099432
E333	10	0.096108	15.63745	0.04098	0.096108
E334	20	0.097309	20	0.147899	0.097309
E335	10	0.096015	17.49309	0.040768	0.096015
E336	10	0.095798	10	0.095798	0.012181
E337	40	0.097276	48.37731	0.040953	0.097276
E338	10	0.095798	15.60812	0.040561	0.095798
E339	10	0.095806	15.58487	0.040334	0.095806
E340	20	0.096512	20	0.096512	0.096512
E341	10	0.095362	17.50091	0.040548	0.095362
E342	20	0.095929	27.40811	0.040098	0.095929
E343	30	0.095611	37.26579	0.038875	0.095611
E344	10	0.095635	17.40811	0.039975	0.095635
E345	60	0.096719	60	0.153327	0.096719
E347	60	0.096762	67.31759	0.03975	0.096762
E348	10	0.095984	15.64927	0.041043	0.095984
E349	10	0.095614	16.4821	0.039967	0.095614
E350	10	0.095184	16.34474	0.038588	0.095184

E351	30	0.095774	36.38339	0.039172	0.095774
E353	40	0.096384	48.24909	0.03971	0.096384
E354	50	0.09613	56.37045	0.039202	0.09613
E355	10	0.095518	15.52193	0.039583	0.095518
E356	70	0.095519	77.22198	0.038494	0.095519
E357	20	0.095095	25.41649	0.038323	0.095095
E358	10	0.095518	15.52759	0.03964	0.095518
E359	10	0.095121	15.42739	0.038448	0.095121
E360	10	0.095341	17.40048	0.039795	0.095341
E361	60	0.098963	67.48528	0.04196	0.098963
E362	10	0.095778	17.46972	0.040495	0.095778
E363	10	0.095422	17.37769	0.039657	0.095422
E364	40	0.09613	40	0.152079	0.09613
E365	60	0.098394	60	0.152926	0.098394
E366	10	0.095798	15.60812	0.040561	0.095798
E367	50	0.096548	50	0.096548	0.096548
E368	10	0.095474	10	0.150729	0.095474
E369	10	0.095093	10	0.151993	0.095093
E370	10	0.095137	15.90346	0.038683	0.095137
E371	60	0.097024	60	0.153176	0.097024
E372	30	0.095698	37.30271	0.039198	0.095698
E374	20	0.095471	25.44935	0.038819	0.095471
E375	50	0.095401	55.41105	0.038389	0.095401
E376	10	0.095123	15.8975	0.03862	0.095123
E377	60	0.096267	66.36401	0.0392	0.096267
E379	30	0.096613	37.37769	0.040152	0.096613
E380	10	0.095299	17.27315	0.038806	0.095299
E382	20	0.094974	25.84436	0.038047	0.094974
E383	10	0.095169	15.46039	0.03881	0.095169
E385	10	0.095484	17.32505	0.039282	0.095484
E386	40	0.096867	48.31698	0.040374	0.096867
E387	10	0.094984	10	0.094984	0.066489
E388	20	0.095202	25.41649	0.038366	0.095202
E389	20	0.095143	25.86197	0.038286	0.095143
E390	10	0.095274	17.35504	0.039424	0.095274
E391	30	0.09528	37.22925	0.038455	0.09528
E392	10	0.095103	10	0.152165	0.095103
E393	60	0.096411	65.45486	0.039259	0.096411
E394	10	0.095494	18.26596	0.039458	0.095494
E395	30	0.096144	30	0.151001	0.096144
E396	10	0.095305	16.34474	0.038637	0.095305
E397	20	0.09542	25.54465	0.039771	0.09542
E398	10	0.094984	15.85609	0.038165	0.094984
E399	40	0.095724	47.26579	0.038921	0.095724
E400	30	0.095209	35.41105	0.038312	0.095209
E401	50	0.096202	50	0.15298	0.096202
E403	50	0.095955	57.26579	0.039015	0.095955
E405	10	0.095264	15.9516	0.039192	0.095264

E406	10	0.095327	15.47704	0.039046	0.095327
E407	10	0.095378	17.39287	0.039754	0.095378
E408	40	0.095741	47.25846	0.038871	0.095741
E409	10	0.095008	17.20027	0.038117	0.095008
E410	20	0.095522	20	0.152054	0.095522
E411	10	0.095184	17.24382	0.03853	0.095184
E412	10	0.094998	15.4002	0.038113	0.094998
E413	40	0.095244	47.20749	0.038269	0.095244
E414	50	0.096629	50	0.096629	0.096629
E415	60	0.095254	65.85609	0.038273	0.095254
E416	60	0.096069	60	0.153239	0.096069
E418	40	0.095644	45.44384	0.038831	0.095644
E421	10	0.095046	17.20027	0.038132	0.095046
E422	10	0.094998	15.3894	0.037999	0.094998
E423	50	0.095347	57.22198	0.038425	0.095347
E425	50	0.095841	55.44935	0.038969	0.095841

附录 C

第一问

1、predeal.py

```
import pandas as pd
import math

data1_in = pd.read_excel('./data1.xlsx', '进项发票信息')
data1_out = pd.read_excel('./data1.xlsx', '销项发票信息')

# 计算月数目
time_a = data1_in.loc[:, ['企业代号', '开票日期']]
time_b = data1_out.loc[:, ['企业代号', '开票日期']]

time_s = pd.concat([time_a, time_b], axis=0)
time_s = time_s.sort_values(by=['企业代号', '开票日期'])
start = time_s.groupby(['企业代号']).first()['开票日期']
end = time_s.groupby(['企业代号']).last()['开票日期']
months = (end-start).dt.days/31
months = months.apply(lambda x: math.ceil(x))


# 1.自身风险部分
# 计算卖出退款率
valid_ticket = data1_out[data1_out['发票状态'] == '有效发票']

sell_fail = valid_ticket[valid_ticket['价税合计'] < 0].groupby('企业代号')[
    '价税合计'].count()
fail_rate = sell_fail / valid_ticket.groupby('企业代号')['价税合计'].count() * 100
fail_rate = fail_rate.fillna(0)


# 卖出取消订单率
invalid_ticket_out = data1_out[data1_out['发票状态'] == '作废发票']

sell_cancel = invalid_ticket_out.groupby('企业代号')['价税合计'].count()
sell_cancel_rate = sell_cancel / \
    data1_out.groupby('企业代号')['价税合计'].count() * 100
sell_cancel_rate = sell_cancel_rate.fillna(0)

# 月均卖出取消订单量
cancel_num_series_out = invalid_ticket_out.groupby(
    ['企业代号'])['价税合计'].count()/months
```

```
# 买入取消订单率
invalid_ticket_in = data1_in[data1_in['发票状态'] == '作废发票']

buy_cancel = invalid_ticket_in.groupby('企业代号')['价税合计'].count()
buy_cancel_rate = buy_cancel / data1_in.groupby('企业代号')['价税合计'].count() *
100
buy_cancel_rate = buy_cancel_rate.fillna(0)

# 月均买入取消订单量
cancel_num_series_in = invalid_ticket_in.groupby(
    ['企业代号'])['价税合计'].count()/months


# 月均买入总交易量
valid_ticket_in = data1_in[data1_in['发票状态'] == '有效发票']
valid_ticket_out = data1_out[data1_out['发票状态'] == '有效发票']

total_in = valid_ticket_in.loc[:, ['企业代号', '金额']].groupby('企业代号').sum()
in_series = valid_ticket_in.loc[:, ['企业代号', '金额']].groupby(['企业代号']).sum()
in_series = in_series.div(months, axis=0)

# 月均卖出总交易量
total_out = valid_ticket_out.loc[:, ['企业代号', '金额']].groupby('企业代号').sum()
out_series = valid_ticket_out.loc[:, ['企业代号', '金额']].groupby(
    ['企业代号']).sum()
out_series = out_series.div(months, axis=0)

# 月均总交易总量
tmp = pd.concat([in_series, out_series], axis=1)
tmp.columns = ['买入', '卖出']
total_in_and_out = tmp['卖出'] + tmp['买入']

money_in = valid_ticket_in.loc[:, ['企业代号', '开票日期', '价税合计']]
money_in['买入'] = money_in['价税合计']

money_out = valid_ticket_out.loc[:, ['企业代号', '开票日期', '金额']]
money_out['卖出'] = money_out['金额']

money_series = pd.concat([money_in, money_out], axis=0,
                         ignore_index=True).fillna(0)
money_series = money_series.sort_values(by=['企业代号', '开票日期'])
money_series['差额'] = money_series['卖出'] - money_series['买入']

# 计算年均净利润
```

```
data_2 = money_series.groupby(['企业代号', money_series['开票日期'].dt.year])['差额'].sum()
data_2 = data_2.groupby(['企业代号']).mean()
data_2.to_excel('./problem1/result/average_income_year.xlsx')
# money_series['计算按年'] = data_2['差额'].cumsum()

# 月净利润
per_income = money_series.groupby(
    ['企业代号'])['差额'].sum()/months
per_income_fangcha = money_series.groupby(['企业代号'])['差额'].std()

# 月均进销项有效发票数
a = valid_ticket_in.groupby(
    ['企业代号'])['开票日期'].count()/months
b = valid_ticket_out.groupby(
    ['企业代号'])['开票日期'].count()/months
tmp = pd.concat([a, b], axis=1)
tmp.columns = ['买入', '卖出']
tmp = tmp.fillna(0)
valid_ticket_series = tmp['卖出'] + tmp['买入']

# 2.上游风险部分
# 上游公司数随每月变化
up_num_series = valid_ticket_in.groupby(['企业代号', valid_ticket_in['开票日期'].dt.year,
                                         valid_ticket_in['开票日期'].dt.month])['销方单位代号'].nunique()

up_cancel = valid_ticket_in[valid_ticket_in['价税合计'] < 0].groupby(['企业代号'])[
    '销方单位代号'].count()/months

v = valid_ticket_in.groupby(['企业代号', '销方单位代号'])['价税合计'].sum()
in_std = v.reset_index().groupby('企业代号').std().fillna(0)


# 3.下游风险部分
down_num_series = valid_ticket_out.groupby(['企业代号', valid_ticket_out['开票日期'].dt.year,
                                            valid_ticket_out['开票日期'].dt.month])['企业代号'].nunique()

down_cancel = valid_ticket_out[valid_ticket_out['价税合计'] < 0].groupby(['企业代号'])[
    '企业代号'].count()
```

```
down_cancel = down_cancel.div(months, axis=0)

o = valid_ticket_out.groupby(['企业代号', '购方单位代号'])['价税合计'].sum()
out_std = o.reset_index().groupby('企业代号').std().fillna(0)


# 4.企业信息处理
data1_info = pd.read_excel('./data1_info.xlsx', '企业信息')

def cal_redit(a, b):
    ll1 = ['D', 'C', 'B', 'A']
    ll2 = ['是', '否']
    return ll1.index(a) * ll2.index(b)


data1_info['信誉度'] = data1_info.apply(
    lambda x: cal_redit(x['信誉评级'], x['是否违约']), axis=1)
credit = data1_info.loc[:, ['企业代号', '信誉度']].set_index('企业代号')

# 5.行业风险计算
category = data1_info['行业'].drop_duplicates().to_numpy()
risk = {}
for i in category:
    ok = data1_info[(data1_info['行业'] == i) & (
        data1_info['是否违约'] == '否')].count()['企业名称']
    total = data1_info[data1_info['行业'] == i].count()['企业名称']
    risk[i] = 1 - ok/total

data1_info['行业违约率'] = data1_info.apply(lambda x: risk[x['行业']], axis=1)
risk_series = data1_info.loc[:, ['企业代号', '行业违约率']].set_index('企业代号')


# 结果生成部分
res = pd.concat([down_cancel, per_income, per_income_fangcha, total_in_and_out,
up_cancel, in_std, cancel_num_series_out,
                 out_std,  credit, risk_series], axis=1)
print(res)
res = (res-res.min())/(res.max()-res.min())
res = res.fillna(0)
res.columns = ['卖出取消次数', '净利润', '净利润方差', '交易总额',
               '买入取消次数',  '上游依赖度', '卖出作废次数', '下游依赖度', '
自身信誉度', '行业违约率']
res['净利润'] = 1-res['净利润']
res['交易总额'] = 1-res['交易总额']
```

```
res['自身信誉度'] = 1-res['自身信誉度']
print(res)
res.to_excel('./problem1/result/problem1_part1.xlsx')
```

2、shang.py

```
import pandas as pd
import numpy as np
from pandas.core.frame import fmt

df = pd.read_excel('./problem1/result/problem1_part1.xlsx', index_col=0)
data = df.iloc[:, [0, 1, 2, 3, 4, 5, 7, 8, 9]].to_numpy()
xishu = pd.read_excel('./problem1/result/lambda.xlsx', index_col=0, header=0)
xishu = xishu.to_numpy()
data0 = np.dot(data, xishu)

# 把结果同向化
data0[:,3] = -data0[:,3]

def entropy(data0):
    # 返回每个样本的指数
    # 样本数，指标个数
    n, m = np.shape(data0)
    # 一行一个样本，一列一个指标
    # 下面是归一化
    maxium = np.max(data0, axis=0)
    minium = np.min(data0, axis=0)
    data = (data0-minium)*1.0/(maxium-minium)
    # 计算第 j 项指标，第 i 个样本占该指标的比重
    sumzb = np.sum(data, axis=0)
    data = data/sumzb
    # 对 ln0 处理
    a = data*1.0
    a[np.where(data == 0)] = 0.0001
    # 计算每个指标的熵
    e = (-1.0/np.log(n))*np.sum(data*np.log(a), axis=0)
    # 计算权重
    w = (1-e)/np.sum(1-e)
    print(w)
    recodes = np.sum(data0*w, axis=1)
    return recodes

res = entropy(data0)
res = (res-np.min(res))/(np.max(res) - np.min(res))
```

```
info = pd.read_excel('./data1_info.xlsx', '企业信息')
info = info.sort_values(['企业代号'])


def deal(rank, ifOff):
    ll1 = ['D', 'A', 'B', 'C']
    ll2 = ['是', '否']
    if(rank == 'D' or ifOff == '是'):
        return -1
    else:
        return ll1.index(rank) * ll2.index(ifOff)


info['类型'] = info.apply(lambda x: deal(x['信誉评级'], x['是否违约']), axis=1)
# info.to_excel('./indexnum.xlsx')
res1 = info['类型'].to_numpy().reshape(-1, 1)


res = np.hstack([res.reshape(-1, 1), res1, data0])
res = pd.DataFrame(res).set_index(info['企业代号'])
res = res[res[1] > 0]
res.to_csv('./problem1/result/risk_rate.csv')
```

3、fit.m

```
clear;
close all;
clc;

data = xlsread('../../data3.xlsx');
x = data(:,1);

figure(1)
hold on;

for i=2:4
    y=data(:,i);
    plot(x,y,'o');

    [parameter,resnorm]=lsqcurvefit(@fun,[0.1,3],x,y)

    xi=0.03:0.001:0.15;
    yi = fun(parameter,xi);
    plot(xi,yi);
end
```

```
xlabel('利率')
ylabel('客户流失率')
```

4. fun.m

```
function f = fun(a,x)
%UNTITLED2  此处显示有关此函数的摘要
%     此处显示详细说明
    f = a(2).*(x-a(1)).^(1/2);
end
```

5. income_old.m

```
function f = income_old(x)
    [m,~] = size(x);

    rate = x(1:m/2, :);
    quota = x(m/2+1:end, :); %每万

    para = csvread('../result/risk_rate.csv',1,1);

    beita = zeros(m/2,1);
    category = para(:,2);
    for i=1:m/2
        if(category(i) == 1)
            beita(i) = 2.856*(rate(i)-0.04)^0.5;
        elseif(category(i) == 2)
            beita(i) = 2.728*(rate(i)-0.04)^0.5;
        elseif(category(i) == 3)
            beita(i) = 2.707*(rate(i)-0.04)^0.5;
        end
    end

    alpha = para(:,1);


    tmp = (alpha .* rate .* quota - (-alpha) .* quota*1000) .* (1-beita*1.8);
    f = -sum(tmp);

end
```

6. predeal.py

```
# 计算平均年利润
import pandas as pd
import numpy as np
```

```
import math

data = pd.read_excel('./problem1/result/average_income_year.xlsx', index_col=0)
info = pd.read_excel('./data1_info.xlsx', '企业信息', index_col=0)

df = pd.concat([info['信誉评级'], info['是否违约'], data], axis=1).sort_index()
df = df[(df['差额'] > 0) & (df['信誉评级'] != 'D')]


res = np.empty([0, 10])
vator = {'A': 0.8, 'B': 1, 'C': 1.2}
changshu = {'A': 2, 'B': 1.8, 'C': 1.6}
p_data={'A': 1, 'B': 0.9, 'C': 0.8}
for index, row in df.iterrows():
    print(index)
    tmp = np.zeros([1, 10])

    category = row['信誉评级']

    index_start = math.floor(row['差额'] * vator[category] // 10e4)
    tmp[0, :index_start] = p_data[category]

    if(index_start > 9):
        res = np.append(res, tmp, axis=0)
        continue
    for j in range(index_start, 10):
        tmp[0, j] = changshu[category] / \
            (1+math.exp((j+1)*10 - vator[category] * row['差额'] / 1e4))
    res = np.append(res, tmp, axis=0)
    print(res)

np.savetxt('./problem1/result/part2_array.csv', res, fmt='%.6f', delimiter=',')
```

7. solve.m

```
clear;
close all;
clc;

para = csvread('../result/risk_rate.csv',1,1);
[m,~] = size(para);

low = ones(1,m*2);
low(1, 1:m) = low(1, 1:m)*0.04;
low(1, m+1:end) = low(1, m+1:end)*1;
```

```
up = ones(1,m*2);
up(1, 1:m) = up(1, 1:m)*0.15;
up(1, m+1:end) = up(1, m+1:end) * 10;

Aeq = ones(1,m*2);
Aeq(1, 1:m) = Aeq(1, 1:m)*0;

% options = optimoptions('fmincon','MaxFunctionEvaluations',4e3)
[x,fval] = fmincon(@income_old, ones(m*2,1)*20,Aeq,100,[],[], low, up,[])
x(m+1:end) = round(x(m+1:end))*10;

csvwrite('../problem1_result.csv',x)
```

第二问

8. label_predit.py

```
# 完成指标恢复工作

import numpy as np
import pandas as pd
from pandas.core.frame import fmt
import sklearn.tree as st
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn import tree

# 这个包含标签是第一问数据用于训练
df = pd.read_excel('./problem2/result/problem2_part1_train.xlsx', index_col=0)
df1 = pd.read_excel('./problem2/result/problem2_part1_getres.xlsx', index_col=0)

data = df.iloc[:, [0,1,2,3,4,5,7,8,9,10]].to_numpy()
y = data[:, 7].astype(int)
y_xinyu=data[:,8].astype(int)
y_if = data[:,9].astype(int)
train_x, test_x, train_y, test_y = train_test_split(data[:,:7], y_if.T, test_size=0.3, random_state=2)

model = st.DecisionTreeClassifier(max_depth=4, criterion='entropy', min_samples_split=8, random_state=2)
# model = MLPClassifier(max_iter = 3000, verbose = True, hidden_layer_sizes=(100,400,400,400), activation='relu')
model.fit(train_x, train_y)
# pred_test_y = model.predict(test_x)
```

```
# print(model.score(test_x, test_y))
# print(classification_report(pred_test_y,test_y))

#  获得待预测数据
data1 = df1.iloc[:, [0,1,2,3,4,5,7]].to_numpy()
pred_y = model.predict(data1)

model.fit(data[:,:7], y_xinyu)
res_xinyu = model.predict(data1)

model = st.DecisionTreeClassifier(max_depth=3, criterion='entropy', min_samples_split=8, random_state=2)
model.fit(data[:,:7], y_if)
print(model.score(data[:,:7], y_if))
res_if = model.predict(data1)

# with open("./jueceshu.dot", 'w') as f:
#     f = tree.export_graphviz(model, out_file = f, class_names=[' 否 ',' 是 '], rounded=True, filled=True, special_characters=True, feature_names=['卖出取消次数', '净利润', '净利润方差', '交易总额', '买入取消次数', '上游依赖度', '下游依赖度'])

res = np.concatenate([pred_y.reshape(-1,1),res_xinyu.reshape(-1,1), res_if.reshape(-1,1)],axis=1)
res = pd.DataFrame(res).set_index(df1.index)
res.to_csv('./problem2/result/predit_label.csv')
```

9. predeal1_concat.py

```
import pandas as pd
import math


# 4.企业信息处理
info = pd.read_csv('./problem2/result/predit_label.csv', header=0, index_col=0)
info.columns=['粗筛','信誉评级','是否违约']

def cal_redit(a, b):
    ll1 = ['D', 'A', 'B', 'C']
    ll2 = ['是', '否']
    if(a==4 or b==1):
        return -1
    return a*(1-b)


info['信誉度'] = info.apply(lambda x: cal_redit(x['信誉评级'], x['是否违约']), axis=1)
```

```
credit = info['信誉度']

part = pd.read_excel('./problem2/result/problem2_part1_getres.xlsx', index_col=0, header=0)

res = pd.concat([part, credit], axis=1)

# 合成最后结果
res.to_excel('./problem2/result/problem2_part1_getres_res.xlsx')
```

10. predeal1_getres.py

```
import pandas as pd
import math

data1_in = pd.read_excel('./data2.xlsx', '进项发票信息')
data1_out = pd.read_excel('./data2.xlsx', '销项发票信息')

# 计算月数目
time_a = data1_in.loc[:, ['企业代号', '开票日期']]
time_b = data1_out.loc[:, ['企业代号', '开票日期']]

time_s = pd.concat([time_a, time_b], axis=0)
time_s = time_s.sort_values(by=['企业代号', '开票日期'])
start = time_s.groupby(['企业代号']).first()['开票日期']
end = time_s.groupby(['企业代号']).last()['开票日期']
months = (end-start).dt.days/31
months = months.apply(lambda x: math.ceil(x))


# 1.自身风险部分
# 计算卖出退款率
valid_ticket = data1_out[data1_out['发票状态'] == '有效发票']

sell_fail = valid_ticket[valid_ticket['价税合计'] < 0].groupby('企业代号')[
    '价税合计'].count()
fail_rate = sell_fail / valid_ticket.groupby('企业代号')['价税合计'].count() * 100
fail_rate = fail_rate.fillna(0)


# 卖出取消订单率
invalid_ticket_out = data1_out[data1_out['发票状态'] == '作废发票']

sell_cancel = invalid_ticket_out.groupby('企业代号')['价税合计'].count()
sell_cancel_rate = sell_cancel / \
    data1_out.groupby('企业代号')['价税合计'].count() * 100
```

```
sell_cancel_rate = sell_cancel_rate.fillna(0)

# 月均卖出取消订单量
cancel_num_series_out = invalid_ticket_out.groupby(
    ['企业代号'])['价税合计'].count()/months

# 买入取消订单率
invalid_ticket_in = data1_in[data1_in['发票状态'] == '作废发票']

buy_cancel = invalid_ticket_in.groupby('企业代号')['价税合计'].count()
buy_cancel_rate = buy_cancel / data1_in.groupby('企业代号')['价税合计'].count() *
100
buy_cancel_rate = buy_cancel_rate.fillna(0)

# 月均买入取消订单量
cancel_num_series_in = invalid_ticket_in.groupby(
    ['企业代号'])['价税合计'].count()/months


# 月均买入总交易量
valid_ticket_in = data1_in[data1_in['发票状态'] == '有效发票']
valid_ticket_out = data1_out[data1_out['发票状态'] == '有效发票']

total_in = valid_ticket_in.loc[:, ['企业代号', '金额']].groupby('企业代号').sum()
in_series = valid_ticket_in.loc[:, ['企业代号', '金额']].groupby(['企业代号']).sum()
in_series = in_series.div(months, axis=0)

# 月均卖出总交易量
total_out = valid_ticket_out.loc[:, ['企业代号', '金额']].groupby('企业代号').sum()
out_series = valid_ticket_out.loc[:, ['企业代号', '金额']].groupby(
    ['企业代号']).sum()
out_series = out_series.div(months, axis=0)

# 月均总交易总量
tmp = pd.concat([in_series, out_series], axis=1)
tmp.columns = ['买入', '卖出']
total_in_and_out = tmp['卖出'] + tmp['买入']

# 净利润随时间变化(暂时无用)
# per_income_series = valid_ticket_out['价税合计'] - valid_ticket_in['价税合计']

money_in = valid_ticket_in.loc[:, ['企业代号', '开票日期', '价税合计']]
money_in['买入'] = money_in['价税合计']
```

```
money_out = valid_ticket_out.loc[:, ['企业代号', '开票日期', '金额']]
money_out['卖出'] = money_out['金额']

money_series = pd.concat([money_in, money_out], axis=0,
                                    ignore_index=True).fillna(0)
money_series = money_series.sort_values(by=['企业代号', '开票日期'])
money_series['差额'] = money_series['卖出'] - money_series['买入']
# data_2 = money_series.groupby(['企业代号', money_series['开票日期'].dt.year])
# money_series['计算按年'] = data_2['差额'].cumsum()

#  月净利润
per_income = money_series.groupby(
      ['企业代号'])['差额'].sum()/months
per_income_fangcha = money_series.groupby(['企业代号'])['差额'].std()

#  月均进销项有效发票数
a = valid_ticket_in.groupby(
      ['企业代号'])['开票日期'].count()/months
b = valid_ticket_out.groupby(
      ['企业代号'])['开票日期'].count()/months
tmp = pd.concat([a, b], axis=1)
tmp.columns = ['买入', '卖出']
tmp = tmp.fillna(0)
valid_ticket_series = tmp['卖出'] + tmp['买入']

# 2.上游风险部分
#  上游公司数随每月变化
up_num_series = valid_ticket_in.groupby(['企业代号', valid_ticket_in['开票日期
'].dt.year,
                                                    valid_ticket_in['开票日期
'].dt.month])['销方单位代号'].nunique()

up_cancel = valid_ticket_in[valid_ticket_in['价税合计'] < 0].groupby(['企业代号'])[
      '销方单位代号'].count()/months

v = valid_ticket_in.groupby(['企业代号', '销方单位代号'])['价税合计'].sum()
in_std = v.reset_index().groupby('企业代号').std().fillna(0)


# 3.下游风险部分
down_num_series = valid_ticket_out.groupby(['企业代号', valid_ticket_out['开票日
期'].dt.year,
                                                    valid_ticket_out['开票日期
'].dt.month])['企业代号'].nunique()
```

```
down_cancel = valid_ticket_out[valid_ticket_out['价税合计'] < 0].groupby(['企业代
号'])[
      '企业代号'].count()
down_cancel = down_cancel.div(months, axis=0)

o = valid_ticket_out.groupby(['企业代号', '购方单位代号'])['价税合计'].sum()
out_std = o.reset_index().groupby('企业代号').std().fillna(0)


# 结果生成部分
res = pd.concat([down_cancel, per_income, per_income_fangcha, total_in_and_out,
up_cancel, in_std, cancel_num_series_out,
                          out_std], axis=1)
res.columns = ['卖出取消次数', '净利润', '净利润方差', '交易总额',
                        '买入取消次数',  '上游依赖度', '卖出作废次数', '下游依赖度']
res = (res-res.min())/(res.max()-res.min())
res = res.fillna(0)
res['净利润'] = 1-res['净利润']
res['交易总额'] = 1-res['交易总额']

# res = pd.concat([res, credit], axis=1)

# 用于给出预测结果
res.to_excel('./problem2/result/problem2_part1_getres.xlsx')
```

11. predeal1_train.m

```
# 第一步生成训练数据集
import pandas as pd
import math

data1_in = pd.read_excel('./data1.xlsx', '进项发票信息')
data1_out = pd.read_excel('./data1.xlsx', '销项发票信息')

# 计算月数目
time_a = data1_in.loc[:, ['企业代号', '开票日期']]
time_b = data1_out.loc[:, ['企业代号', '开票日期']]

time_s = pd.concat([time_a, time_b], axis=0)
time_s = time_s.sort_values(by=['企业代号', '开票日期'])
start = time_s.groupby(['企业代号']).first()['开票日期']
end = time_s.groupby(['企业代号']).last()['开票日期']
months = (end-start).dt.days/31
months = months.apply(lambda x: math.ceil(x))
```

```
# 1.自身风险部分
# 计算卖出退款率
valid_ticket = data1_out[data1_out['发票状态'] == '有效发票']

sell_fail = valid_ticket[valid_ticket['价税合计'] < 0].groupby('企业代号')[
    '价税合计'].count()
fail_rate = sell_fail / valid_ticket.groupby('企业代号')['价税合计'].count() * 100
fail_rate = fail_rate.fillna(0)


# 卖出取消订单率
invalid_ticket_out = data1_out[data1_out['发票状态'] == '作废发票']

sell_cancel = invalid_ticket_out.groupby('企业代号')['价税合计'].count()
sell_cancel_rate = sell_cancel / \
    data1_out.groupby('企业代号')['价税合计'].count() * 100
sell_cancel_rate = sell_cancel_rate.fillna(0)

# 月均卖出取消订单量
cancel_num_series_out = invalid_ticket_out.groupby(
    ['企业代号'])['价税合计'].count()/months

# 买入取消订单率
invalid_ticket_in = data1_in[data1_in['发票状态'] == '作废发票']

buy_cancel = invalid_ticket_in.groupby('企业代号')['价税合计'].count()
buy_cancel_rate = buy_cancel / data1_in.groupby('企业代号')['价税合计'].count() *
100
buy_cancel_rate = buy_cancel_rate.fillna(0)

# 月均买入取消订单量
cancel_num_series_in = invalid_ticket_in.groupby(
    ['企业代号'])['价税合计'].count()/months


# 月均买入总交易量
valid_ticket_in = data1_in[data1_in['发票状态'] == '有效发票']
valid_ticket_out = data1_out[data1_out['发票状态'] == '有效发票']

total_in = valid_ticket_in.loc[:, ['企业代号', '金额']].groupby('企业代号').sum()
in_series = valid_ticket_in.loc[:, ['企业代号', '金额']].groupby(['企业代号']).sum()
in_series = in_series.div(months, axis=0)
```

```
# 月均卖出总交易量
total_out = valid_ticket_out.loc[:, ['企业代号', '金额']].groupby('企业代号').sum()
out_series = valid_ticket_out.loc[:, ['企业代号', '金额']].groupby(
    ['企业代号']).sum()
out_series = out_series.div(months, axis=0)

# 月均总交易总量
tmp = pd.concat([in_series, out_series], axis=1)
tmp.columns = ['买入', '卖出']
total_in_and_out = tmp['卖出'] + tmp['买入']

# 净利润随时间变化(暂时无用)
# per_income_series = valid_ticket_out['价税合计'] - valid_ticket_in['价税合计']

money_in = valid_ticket_in.loc[:, ['企业代号', '开票日期', '价税合计']]
money_in['买入'] = money_in['价税合计']

money_out = valid_ticket_out.loc[:, ['企业代号', '开票日期', '金额']]
money_out['卖出'] = money_out['金额']

money_series = pd.concat([money_in, money_out], axis=0,
                                    ignore_index=True).fillna(0)
money_series = money_series.sort_values(by=['企业代号', '开票日期'])
money_series['差额'] = money_series['卖出'] - money_series['买入']
# data_2 = money_series.groupby(['企业代号', money_series['开票日期'].dt.year])
# money_series['计算按年'] = data_2['差额'].cumsum()

# 月净利润
per_income = money_series.groupby(
    ['企业代号'])['差额'].sum()/months
per_income_fangcha = money_series.groupby(['企业代号'])['差额'].std()

# 月均进销项有效发票数
a = valid_ticket_in.groupby(
    ['企业代号'])['开票日期'].count()/months
b = valid_ticket_out.groupby(
    ['企业代号'])['开票日期'].count()/months
tmp = pd.concat([a, b], axis=1)
tmp.columns = ['买入', '卖出']
tmp = tmp.fillna(0)
valid_ticket_series = tmp['卖出'] + tmp['买入']

# 2.上游风险部分
```

```
# 上游公司数随每月变化
up_num_series = valid_ticket_in.groupby(['企业代号', valid_ticket_in['开票日期
'].dt.year,
                                        valid_ticket_in['开票日期
'].dt.month])['销方单位代号'].nunique()

up_cancel = valid_ticket_in[valid_ticket_in['价税合计'] < 0].groupby(['企业代号'])[
    '销方单位代号'].count()/months

v = valid_ticket_in.groupby(['企业代号', '销方单位代号'])['价税合计'].sum()
in_std = v.reset_index().groupby('企业代号').std().fillna(0)


# 3.下游风险部分
down_num_series = valid_ticket_out.groupby(['企业代号', valid_ticket_out['开票日
期'].dt.year,
                                           valid_ticket_out['开票日期
'].dt.month])['企业代号'].nunique()

down_cancel = valid_ticket_out[valid_ticket_out['价税合计'] < 0].groupby(['企业代
号'])[
    '企业代号'].count()
down_cancel = down_cancel.div(months, axis=0)

o = valid_ticket_out.groupby(['企业代号', '购方单位代号'])['价税合计'].sum()
out_std = o.reset_index().groupby('企业代号').std().fillna(0)

# 获得企业评级标签

info = pd.read_excel('./data1_info.xlsx', '企业信息', index_col=0)
info = info.sort_index()


def deal(rank, ifOff):
    ll1 = ['D', 'A', 'B', 'C']
    ll2 = ['是', '否']
    if(rank == 'D' or ifOff == '是'):
        return 4
    else:
        return ll1.index(rank) * ll2.index(ifOff)


info['类型'] = info.apply(lambda x: deal(x['信誉评级'], x['是否违约']), axis=1)
label = info['类型']
```

```
rank = ['', 'A', 'B', 'C', 'D']
ifOff = ['否', '是']
info['信誉评级'] = info.apply(lambda x: rank.index(x['信誉评级']), axis=1)
info['是否违约'] = info.apply(lambda x: ifOff.index(x['是否违约']), axis=1)


# 结果生成部分
res = pd.concat([down_cancel, per_income, per_income_fangcha, total_in_and_out,
up_cancel, in_std, cancel_num_series_out,
                         out_std], axis=1)
res = (res-res.min())/(res.max()-res.min())
res = res.fillna(0)
res.columns = ['卖出取消次数', '净利润', '净利润方差', '交易总额',
                      '买入取消次数',  '上游依赖度', '卖出作废次数', '下游依赖度']
res['净利润'] = 1-res['净利润']
res['交易总额'] = 1-res['交易总额']

# 把标签输出出来
res = pd.concat([res, label, info['信誉评级'], info['是否违约']], axis=1)

res.to_excel('./problem2/result/problem2_part1_train.xlsx')
```

12. predeal2.m

```
# 提取时间序列信息
import numpy as np
import pandas as pd
import math

data1_in = pd.read_excel('./data2.xlsx', '进项发票信息')
data1_out = pd.read_excel('./data2.xlsx', '销项发票信息')

# 月均买入总交易量
valid_ticket_in = data1_in[data1_in['发票状态'] == '有效发票']
valid_ticket_out = data1_out[data1_out['发票状态'] == '有效发票']

money_in = valid_ticket_in.loc[:, ['企业代号', '开票日期', '价税合计']]
money_in['买入'] = money_in['价税合计']

money_out = valid_ticket_out.loc[:, ['企业代号', '开票日期', '金额']]
money_out['卖出'] = money_out['金额']

money_series = pd.concat([money_in, money_out], axis=0,
```

```
                                    ignore_index=True).fillna(0)
money_series = money_series.sort_values(by=['企业代号', '开票日期'])
money_series['差额'] = money_series['卖出'] - money_series['买入']

money_series = money_series[money_series['开票日期'].dt.year == 2019]
per_income_series = money_series.groupby(['企业代号', money_series['开票日期'].dt.month])['差额'].sum()
# per_income_series.to_excel('./tmp.xlsx')


index=[]
for i in set(money_series['企业代号']):
    for month in range(1,13):
        index.append((i,month))
series = pd.DataFrame(index= pd.Index(index,names=["企业代号","开票日期"]))

res = pd.concat([per_income_series, series], axis=1).fillna(0).reset_index()


res_list={}
for i in set(money_series['企业代号']):
    data = res[res['企业代号']==i].to_numpy()[:,2]
    for k in range(1,7):
        x = data[:(12-k)]
        y = data[k:]
        x_ba = np.mean(x)
        y_ba = np.mean(y)
        r = np.sum((x - x_ba)*(y - y_ba)) / np.sqrt(np.sum(np.square(x-x_ba))) / np.sqrt(np.sum(np.square(y-y_ba)))
        # print(r)

        if(r>0.6):
            res_list[i] = k
            break
    else:
        res_list[i] = -2

df = pd.read_csv('./problem2/result/risk_rate.csv',header=0, index_col=0)

def deal(index):
    return res_list[index]/2

df['周期']=df.apply(lambda x: deal(x.name), axis=1)
df.to_csv('./problem2/result/risk_rate_add.csv')
```

13. shang.py

```
import pandas as pd
import numpy as np

df = pd.read_excel('./problem2/result/problem2_part1_getres_res.xlsx', index_col=0)
data = df.iloc[:, [0, 1, 2, 3, 4, 6, 7]].to_numpy()
xishu = pd.read_excel('./problem2/result/lambda.xlsx', index_col=0, header=0)
xishu = xishu.to_numpy()
data0 = np.dot(data, xishu)


def entropy(data0):
    # 返回每个样本的指数
    # 样本数，指标个数
    n, m = np.shape(data0)
    # 一行一个样本，一列一个指标
    # 下面是归一化
    maxium = np.max(data0, axis=0)
    minium = np.min(data0, axis=0)
    data = (data0-minium)*1.0/(maxium-minium)
    # 计算第 j 项指标，第 i 个样本占该指标的比重
    sumzb = np.sum(data, axis=0)
    data = data/sumzb
    # 对 ln0 处理
    a = data*1.0
    a[np.where(data == 0)] = 0.0001
    # 计算每个指标的熵
    e = (-1.0/np.log(n))*np.sum(data*np.log(a), axis=0)
    # 计算权重
    w = (1-e)/np.sum(1-e)
    print(w)
    recodes = np.sum(data0*w, axis=1)
    return recodes

res = entropy(data0)
res = (res-np.min(res))/(np.max(res) - np.min(res))

# 4.企业信息处理
info = pd.read_csv('./problem2/result/predit_label.csv', header=0, index_col=0)
info.columns=['粗筛','信誉评级','是否违约']

def cal_redit(a, b):
    ll1 = ['D', 'A', 'B', 'C']
```

```
    ll2 = ['是', '否']
    if(a==4 or b==1):
        return -1
    return a*(1-b)


info['信誉度'] = info.apply(lambda x: cal_redit(x['信誉评级'], x['是否违约']), axis=1)
credit = info['信誉度'].to_numpy().reshape(-1, 1)

res = np.hstack([res.reshape(-1, 1),credit,data0])

res=pd.DataFrame(res).set_index(df.index)
res = res[res[1] > 0]
res.to_csv('./problem2/result/risk_rate.csv')


# np.savetxt('./problem2/result/risk_rate.csv', res, delimiter=',', fmt='%.3f')
```

14. income_old.m

```
function f = income_old(x)
    [m,~] = size(x);
    m=m-1;

    rate = x(1:m/2, :);
    quota = round(x(m/2+1:end-1, :));    % 每 10 万

    para = csvread('../result/risk_rate_add.csv',1,1);

    beita = zeros(m/2,1);
    category = para(:,2);
    for i=1:m/2
        if(category(i) == 1)
            beita(i) = 2.856*(rate(i)-0.04)^0.5;
        elseif(category(i) == 2)
            beita(i) = 2.728*(rate(i)-0.04)^0.5;
        elseif(category(i) == 3)
            beita(i) = 2.707*(rate(i)-0.04)^0.5;
        end
    end
    alpha = para(1:end-1,1);

    res=0;
    for i=1:m/2-1
        if para(i,6)==-1
```

```
                tmp = (alpha(i) * rate(i) *beita(i) - (1-alpha(i)) * quota(i)*1000 ) *
(1-beita(i)*1.8);
            else
                t0 = para(i,6)/12;
                t1 = 1-para(i,6)/12;
                tmp = t0*(alpha(i) .* rate(i) .*beita(i) .* x(m+1) - (1-alpha(i)) .*
quota(i) * 1000 ) * (1-beita(i)*1.8) + t1.*(alpha(i) .* rate(i) .*beita(i) - (1-alpha(i)) *
quota(i)*1000) * (1-beita(i)*1.8);
            end
            res = res+tmp;
        end

%         for i=1:m/2-1
%             tmp = (alpha(i) .* rate(i) .* quota(i) - (1-alpha(i)) .* quota(i) *1000) .*
(1-beita(i)*1.8);
%             res = res+tmp;
%         end

        f = -res;

end
```

15. solve.m

```
clear;
close all;
clc;

para = csvread('../result/risk_rate_add.csv',1,1);
[m,~] = size(para);

low = ones(1,m*2+1);
low(1, 1:m) = low(1, 1:m)*0.04;
low(1, m+1:end-1) = low(1, m+1:end-1)*1;
low(1, end) = 0;

up = ones(1,m*2+1);
up(1, 1:m) = up(1, 1:m)*0.15;
up(1, m+1:end-1) = up(1, m+1:end-1) * 10;

Aeq = ones(1,m*2+1);
Aeq(1, 1:m) = Aeq(1, 1:m)*0;
Aeq(1, m*2+1) = 0;

options = optimoptions('fmincon','MaxFunctionEvaluations',8e4,'StepTolerance',
1e-3);
```

```
[x,fval] = fmincon(@income_old, rand(m*2+1,1)*10,[],[],Aeq,1000, low, up,[])
x(m+1:end-1) = round(x(m+1:end-1))*10;
sum(x(m+1:end-1))

csvwrite('../result/problem2_result.csv',x)
```

16.solve.m

```
import pandas as pd

df = pd.read_excel('./data2_info.xlsx', '企业信息', index_col=0)
para = pd.read_excel('./problem3/risk_rate.xlsx', header=0, index_col=0)

categorys = ['工业', '文化传媒', '商业', '生活服务业', '个体户']
k = {'工业': 0.8, '文化传媒': 0.7, '商业': 0.6, '生活服务业': 0.65, '个体户': 0.9}

para['风险系数'] = 1 - para['风险系数']
para['addition_q'] = 0
for cate in categorys:
    indexs = set(df[df['行业'] == cate].index) & set(para.index)
    for j in indexs:
        if(para['风险系数'][j] == 0):
            para.loc[j, 'addition_q'] = 0
        else:
            para.loc[j, 'addition_q'] = k[cate] / para['风险系数'][j]
total = sum(para['addition_q'])
para['补偿额度'] = para.apply(lambda x: x['addition_q']/total*2000, axis=1)
para['总额度'] = para['补偿额度'] + para['原始额度']

cate1 = df[df['行业'].isin(['技术行业'])]
cate1_index = cate1.index
cate2 = df[df['行业'].isin(['工业', '文化传媒', '商业', '生活服务业', '个体户'])]
cate2_index = cate2.index
cate3 = df[df['行业'].isin(['农林牧渔', '邮政物流'])]
cate3_index = cate3.index

r_cate2 = para[para.index.isin(cate2_index)]['风险系数']
r_cate1 = para[para.index.isin(cate1_index)]['风险系数']


def deal(index, ri, v):
    if (index in cate1_index):
        g = (ri - min(r_cate1))/(max(r_cate1) - min(r_cate1))
        beita = 0.6
        return v*(1+beita*g)
```

```
    elif(index in cate2_index):
        g = (ri - min(r_cate2))/(max(r_cate2) - min(r_cate2))
        alpha = 0.6
        return v*(1-alpha*g)
    else:
        return v

para['新利率'] = para.apply(lambda x: deal(x.name, x['风险系数'], x['原始利率']),
axis=1)


level1 = para.loc[['E185', 'E336','E179','E267', 'E329'], :]['风险系数']
level2 = para.loc[['E141', 'E308','E308','E387'],:]['风险系数']
level_total = pd.concat([level1,level2], axis=0)


def deal_1(index, ri, v):
    g = (ri - min(level_total))/(max(level_total) - min(level_total))
    if (index in level1.index):
        return v*(1-g)
    elif(index in level2.index):
        return v*(1-0.3*g)
    else:
        return v
para['旱灾利率'] = para.apply(lambda x: deal_1(x.name, x['风险系数'], x['原始利率
']), axis=1)

para.to_excel('./problem3/调整后结果.xlsx')
```

17. analyze.py

```
# 计算平均年利润
import pandas as pd
import numpy as np
import math

data = pd.read_excel('./problem1/result/average_income_year.xlsx', index_col=0)
info = pd.read_excel('./data1_info.xlsx', '企业信息', index_col=0)

df = pd.concat([info['信誉评级'], info['是否违约'], data], axis=1).sort_index()
df = df[(df['差额'] > 0) & (df['信誉评级'] != 'D')]

bianhua=[]
for a in list(range(105,130,1)):
    tmp=a/100
```

```
    res = np.empty([0, 10])
    vator = {'A': 0.8, 'B': 1, 'C': tmp}
    changshu = {'A': 2, 'B': 1.8, 'C': 1.6}
    p_data={'A': 1, 'B': 0.9, 'C': 0.8}
    for index, row in df.iterrows():
        print(index)
        tmp = np.zeros([1, 10])

        category = row['信誉评级']

        index_start = math.floor(row['差额'] * vator[category] // 10e4)
        tmp[0, :index_start] = p_data[category]

        if(index_start > 9):
            res = np.append(res, tmp, axis=0)
            continue
        for j in range(index_start, 10):
            tmp[0, j] = changshu[category] / \
                (1+math.exp((j+1)*10 - vator[category] * row['差额'] / 1e4))
        res = np.append(res, tmp, axis=0)
    bianhua.append(np.mean(res))

print(bianhua)
```

18. paint.m

```
close all;
clear;
clc;


x1 = 0.5:0.01:0.89;
a = [0.6920469468511322, 0.69275371714987, 0.6927594863802926,
0.6927693834977948, 0.6927898201291353, 0.692842728321061,
0.6930021427317276, 0.6934553206953243, 0.6941700894642736,
0.6942902448077999, 0.6944969439615537, 0.6948478448738564,
0.6954091460884814, 0.6962232486221188, 0.6968868831362844,
0.697964857822913, 0.6981616430943253, 0.6982829945067327,
0.6984284102083084, 0.6986087930018781, 0.6988648989403013,
0.6993266497275344, 0.7000497408385714, 0.7004432332818733,
0.7007060578840673, 0.7011133376089899, 0.7014438270515989,
0.7014637785076459, 0.7014916819410004, 0.7015312452468991,
0.7015879949884647, 0.701670069014363, 0.701789195071603,
0.7019617708862053, 0.7022099558127177, 0.7025626551840265,
```

```
0.703054436499244,  0.7036697629342044,  0.7041882755555979,
0.7044077707470092];


x2=0.85:0.01:1.09;
b=[0.6864041438802001,  0.6874082220665664,  0.6887463106067377,
0.6898205123571999,  0.6907446829973849,  0.6920088773505211,
0.6931644352597656,  0.6946986614853583,  0.695708440542146,
0.6964340959406362,  0.6970410710118091,  0.6974925534777611,
0.6982017178954385,  0.6992743904241886,  0.70052080435794,
0.7015879949884647,  0.7022151797270498,  0.7029713414315406,
0.7038720738402258,  0.7045602255352664,  0.7051210463477269,
0.7060244636123337,  0.7073415952842919,  0.7087811226526635,
0.7092770018346319];

x3 = 1.05:0.01:1.29;
c=[0.6921010370553013,  0.6925826826510836,  0.6933142298411196,
0.6938750927310793,  0.6943385269150235,  0.6948484345219362,
0.6956515035589087,  0.696821659368745,  0.6979575649326037,
0.6987974335463585,  0.6995068033679849,  0.7002896997445103,
0.7010277308800584,  0.7014045198788558,  0.7015251085743164,
0.7015879949884647,  0.7016796210606184,  0.7018166577094569,
0.7020248803715352,  0.7023410910220143,  0.7028105539479291,
0.7034736567479453,  0.7042362475833123,  0.704970985668506,
0.7052480904941013];


figure(1)
plot(x1, a,'LineWidth',2);
set(gca,'FontSize',18)
xlabel('\omega_A')
ylabel('$\bar{P}$','Interpreter','LaTex')
figure(2)
plot(x2, b,'LineWidth',2);
set(gca,'FontSize',18)
xlabel('\omega_B')
ylabel('$\bar{P}$','Interpreter','LaTex')
figure(3)
plot(x3,c,'LineWidth',2)
set(gca,'FontSize',18)
xlabel('\omega_C')
ylabel('$\bar{P}$','Interpreter','LaTex')
```