中小微企业的信贷决策

摘要

本文针对银行对小微企业的信贷策略展开研究，通过调查研究影响银行贷款风险的各个因素，建立银行信贷风险与企业各项指标间关系的数学模型，并将各个指标进行量化计算，进而为企业科学的信贷策略的制定提供合理科学的参考。本文通过建立一个最为核心的“影响因素层级图”并据此建立各个模型；利用 python 的 Numpy、pandas 库对附件中的数据进行挖掘、分析；matplotlib 库绘制相关图形；利用 Spssau 在线网站对数据进行各项科学分析，进而求解各个模型。

问题一中，重点在于建立模型合理评估企业信贷风险以及量化计算各个指标。本文建立“影响因素层级图”，并据此确定了各个指标的相互依赖关系，进一步逐层地建立数学模型。对于信贷风险的量化计算，本文采用了熵权法；并且对于影响信贷风险的三个子指标采用 AHP 层次分析法进行量化计算。对于最底层的一些指标，如：信誉评级、是否违约等，本文都对它们进行了归一化处理，以便于后续计算。最后，根据企业的信贷风险和附件 3 中的信息建立银行贷款策略模型并求解。

问题二中，由于缺少信贷记录，而无法直接使用问题一中的模型。需要先根据月均订单量、月均利率、销进作废占比等信息建立信誉预测模型，采用二元 logit 回归对各指标进行回归分析求解得出回归方程，回归方程用于求解企业的违规概率，通过企业的违约概率来衡量企业的信誉评级，进而求出信誉值。之后再使用问题一中的模型给出贷款策略建议。

问题三中，新增影响指标：突发因素。本文认为突发因素表现在对企业的实力的影响，进而影响了信贷风险。于是我们以新冠疫情为例，收集各行业受到的影响数据，量化该指标；并且对所有的企业分类，用“对行业的影响”近似估计“对具体企业的影响”，从而求出新的企业实力并代入前面的模型。

关键词：AHP 层次分析；拟合；熵值法；logit 回归分析；信贷策略

一　问题重述

1.1 问题背景

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。

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。附件 1~3 分别给出了 123 家有信贷记录企业的相关数据、302 家无信贷记录企业的相关数据和贷款利率与客户流失率关系的 2019 年统计数据。该银行请你们团队根据实际和附件中的数据信息，通过建立数学模型研究对中小微企业的信贷策略。

1.2 求解问题

本文作者依据附件中的大量数据，并利用 python 中的 Numpy、pandas 等工具库对其进行数据挖掘和数据分析，并建立数学模型解决了以下问题：

(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。

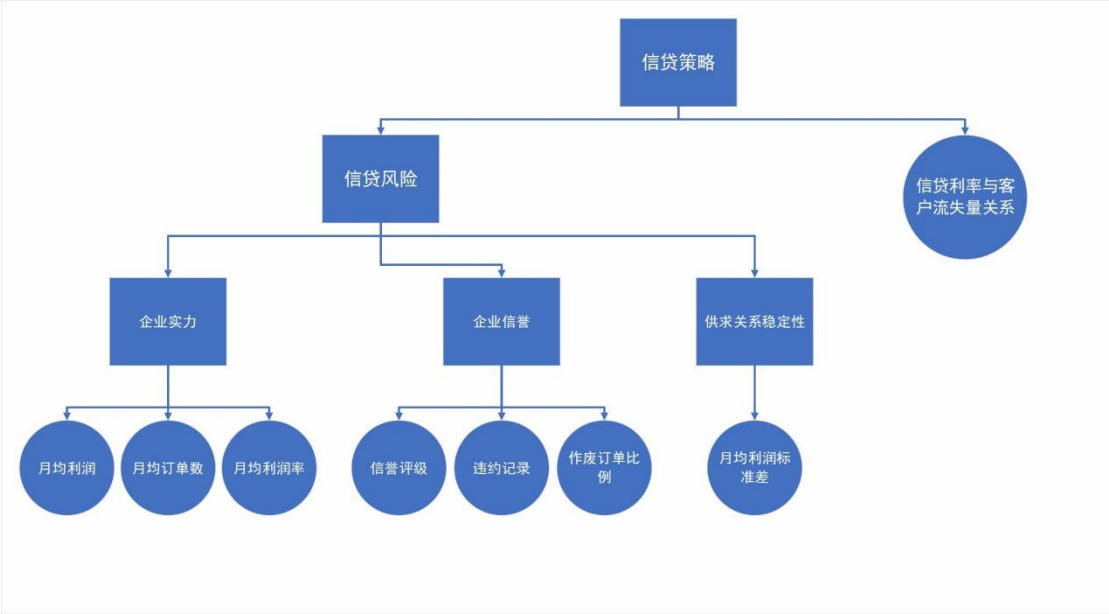
(2) 在问题 1 的基础上，对附件 2 中 302 家企业的信贷风险进行量化分析，并给出该银行在年度信贷总额为 1 亿元时对这些企业的信贷策略。

(3) 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。综合考虑附件 2 中各企业的信贷风险和可能的突发因素（例如：新冠病毒疫情）对各企业的影响，给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

2.1 问题一的分析

根据题目的要求以及附件中所提供的信息，可以发现此问题中的影响因素较多且具有明显的层次机构，较高层的因素由较低层的因素决定，而最底层的因素，如：企业的信誉评级、企业的销售额等则来自于附件。因此，本文为此问题设计了“影响因素层级图”



图表 1 影响因素层级图

图中圆框内的因素便可以从附件中通过挖掘、分析来得到。因此需要逐层建立数学模型来量化地求出各个企业的信贷风险值，并据此制订银行的信贷策略。同时，由于此问题数据量非常大，因此本文选择 python 中的 Numpy 和 pandas 库进行数据筛选与分析，并实现各个影响因素的量化。

此外，由于此问题没有统一的量纲，故本文对上述层级图中涉及的全部影响因素都进行“归一化”量化处理，以方便比较与计算，归一化公式：

$$x = \frac{x - \min}{\max - \min}$$

2.2 问题二的分析

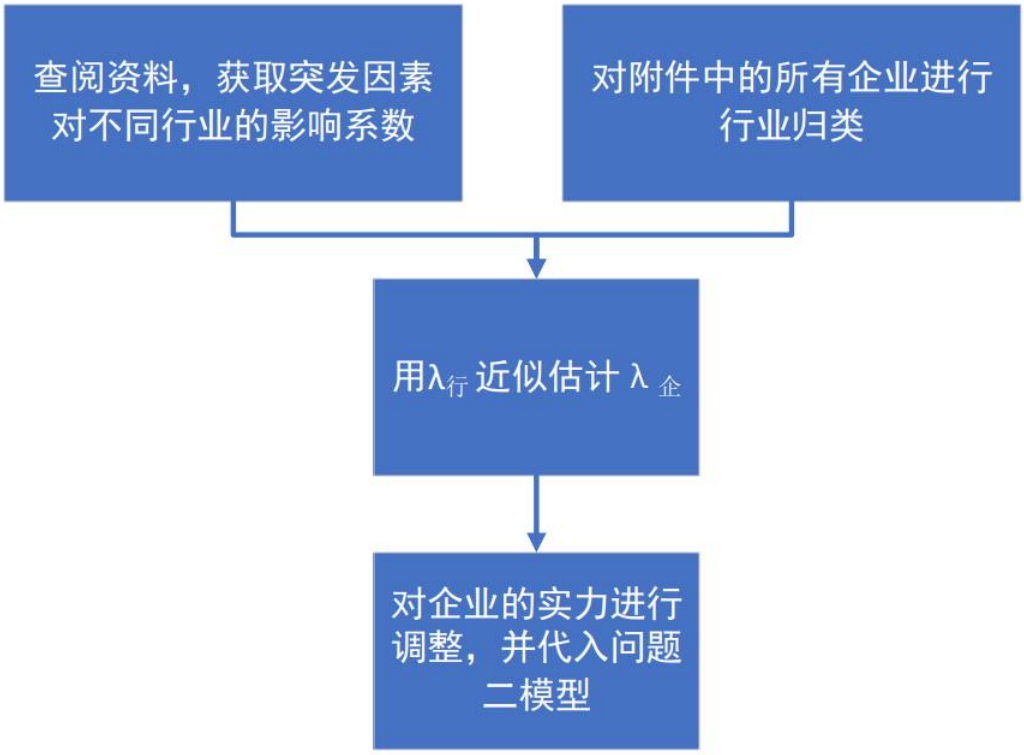
第二问比较第一问差别在于无企业信贷记录的缺少。需要依据其他变量来合理的预测无信贷记录的企业的信誉值。

依据附件 1 中的数据，对违约与否映射为 1 和 0，再对月均订单数、销进作废发票占比、销进负单占比、月均利润和月均利润额进行回归分析，从而求得回归方程用于预测各个企业的违约率，再根据违约率预测信誉评级，得到这些数据之后再套用问题一中的信誉评价模型算出信誉值，进而可以利用信贷风险评价模型得出信贷风险值，给出信贷策略建议

2.3 问题三的分析

此问题其实就是在问题二的基础上对模型进行调整，再多考虑一个影响因素：“突发因素对各企业的影响系数”，记做 $\lambda_{企}$。本文假定 $\lambda_{企}$ 只影响企业实力 P，据此求出 $P_{新}$。再将 $P_{新}$ 代入问题二的模型中即可得解。

考虑到企业的代号与名称繁杂，直接处理 $\lambda_{企}$ 很困难。于是借助概率论与数理统计的知识，本文使用“突发因素对各行业的影响系数 $\lambda_{行}$ ”来近似估计 $\lambda_{企}$。思路图如下



图表 2 问题三的思路

三 问题假设

1. 假设附件 1 中数据具有一般代表性
2. 假设企业交易发票真实有效
3. 假设各个信誉等级的企业占比反应一般水平
4. 假设在突发因素发生后，本行业的总体利润率短期内波动，长远来看趋向正常
5. 假设企业实力和企业信誉之间没有必然关系

四　符号说明

注：下表中所有需要“归一化”的变量都会进行“归一化”处理

变量	说明
Y	银行信贷策略
F	企业信贷风险值
S	企业供求稳定性值
C	企业信誉值
C_1	信誉评级值
C_2	违约记录值
C_3	作废发票比值
P	企业实力值
P_1	月均利润值
P_2	月均发票值
P_3	月均利润率
I_j	熵权法中第 j 个指标的熵值
w_j	熵权法中第 j 个指标的熵权
X	熵权法中的初始矩阵
α_j	AHP 中第 j 个指标的权重
$G(x)$	对 x 进行归一化处理后的值
R_A，R_B，R_C	信誉等级分别为 A,B,C 的企业的贷款利率
T_A，T_B，T_C	信誉等级分别为 A,B,C 的客户流失率
Z_{Ei}	企业代号为 Ei 的公司可贷款额度
M	银行贷款收益

$\lambda_{企}$	突发因素对企业的影响系数
$\lambda_{行}$	突发因素对行业的影响系数

五　模型建立与求解

5.1 问题一

5.1.1 企业信誉评价模型

(1)建立

在本文设计的“影响因素层级图中”，企业信誉C受“信誉评级”、“违约记录”和“作废发票占比”影响，分别记为C_1,C_2,C_3，且这三个子因素对C的影响程度明显不同，且彼此之间的相对重要性也不同，因此本文运用AHP层次分析法将定量分析与定性分析结合起来，通过请教相关领域的专家的经验判断C_i之间的相对重要程度，并合理地给出C_1,C_2,C_3对C的影响权重α_i，从而求得

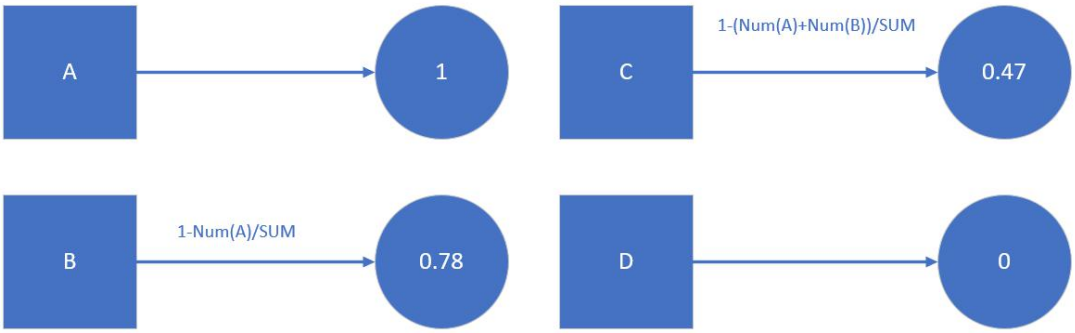
$$C=\alpha_1C_1+\alpha_2C_2+\alpha_3C_3$$

此模型利用权重求出C_i的优劣次序，能够有效地用定量方法求出企业的信誉值。

(2)数据处理

Step1

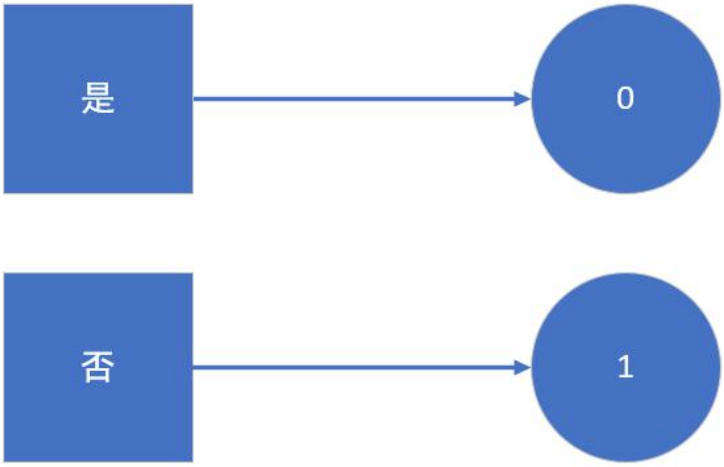
附件1中给出了对各个企业的信誉评级，本文依据各个等级所占的百分比构造一个映射，从而完成对信誉评级C_1的量化



图表 3 信誉评级量化映射

Step2

对于违约记录C_2,因为它与企业信誉值C是负相关，故做映射



图表 4 违约记录量化映射

Step3

本文认为企业的作废发票比率与企业信誉值负相关，故

$$C_3 = G(1 - rate_{废})$$

G表示归一化处理

(3)求解

Step1

本文咨询了 10 位领域内专家，确立了此三个因素的相互重要性矩阵

	违约	评级	作废比
违约	1	2	5
评级	0.5	1	4
作废比	0.2	0.25	1

图表 5 信誉影响因素相互重要性矩阵

Step2

利用 SPSSAU 数据科学在线网站进行“AHP 层次分析”，即可求得$\alpha_1,\alpha_2,\alpha_3$

AHP层次分析结果				
项	特征向量	权重值	最大特征值	CI值
违约	1.704	56.787%	3.025	0.012
评级	1.002	33.394%		
作废比	0.295	9.819%		

图表 6 信誉影响因素 AHP 分析

同时进行重要性矩阵一致性检测

一致性检验结果汇总				
最大特征根	CI值	RI值	CR值	一致性检验结果
3.025	0.012	0.520	0.024	通过

图表 7 矩阵一致性检验结果

Step3

利用 python 中的 Numpy 和 pandas 库编程计算出某一个企业信誉值

$$C=\alpha_1C_1+\alpha_2C_2+\alpha_3C_3$$

最后对C进行归一化处理即可

$$C = G(C)$$

G为归一化处理

5.1.2 企业实力评价模型

(1)建立与求解方法

在本文设计的“影响因素层级图中”，企业实力P受P_1, P_2, P_3影响，同样运用运用 AHP 层次分析法即可求出

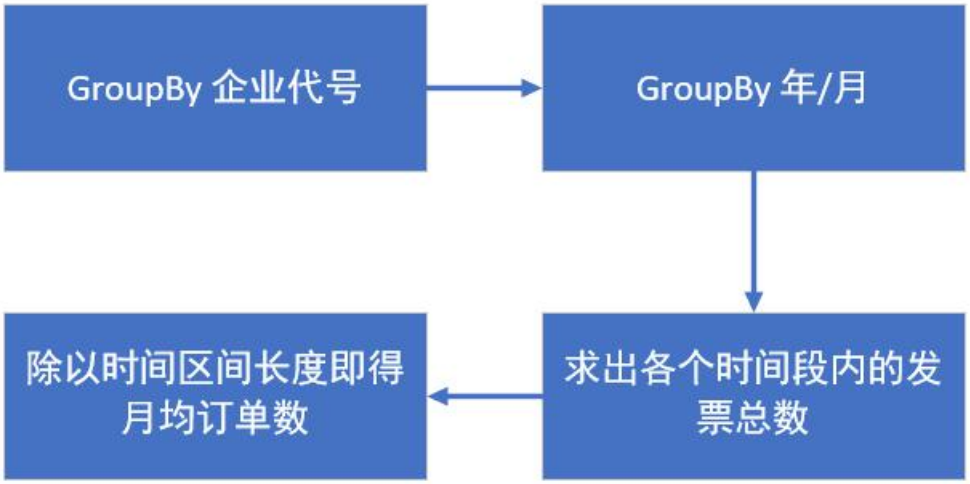
$$P = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3$$

此模型与前文相同，不多赘述。

(2)数据处理

此模型的困难之处在于数据处理。衡量企业实力的主要因素有三大因素：月均订单数，月均利润及月均利润率。在 python 中使用 pandas 库对数据进行筛选及处理

Step1



图表 8 计算月均订单步骤

Step2 & Step3

同上步骤即可

5.1.3 企业供求稳定性评价模型

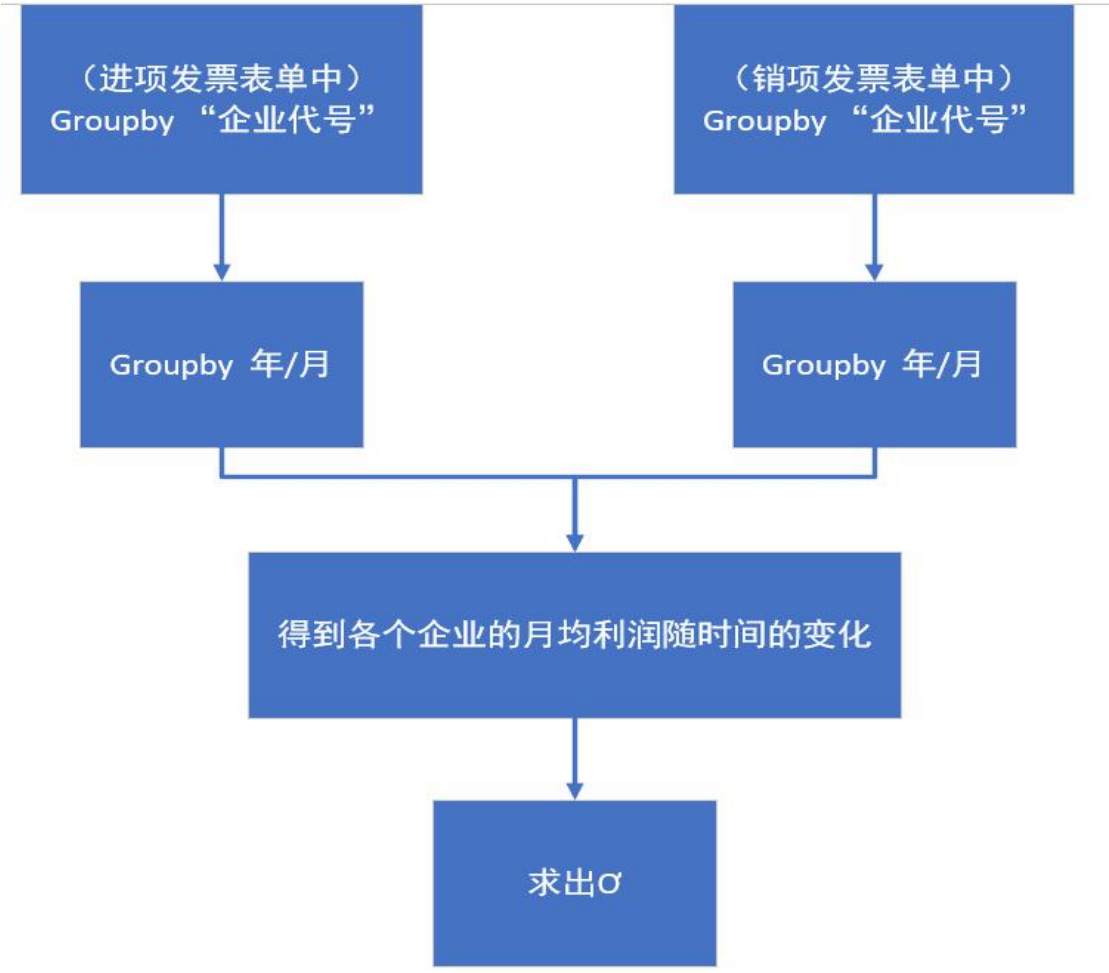
(1)建立

考虑到企业的供求稳定性也是影响信贷风险的重要因素之一，而常用的统计量“标准差”正好可以反应数据的波动情况；此外，月均利润能够很好地反应企业的收益情况。综上，本文使用“企业的月均利润标准差”σ作为衡量“企业供求稳定性”S的指标，且二者成负相关，故

$$S = G(1-\frac{\sigma}{\mathrm{Sum}(\sigma)})$$

G表示归一化处理

(2)数据处理



图表 9 求月均利润标准差

(3)求解

对经过处理过后的销项和进项数据，用销项的价税合计—进项的价税合计得出利润一列，再对利润一列使用 numpy.std()函数求出标准差，日期单位为月。

5.1.4 企业信贷风险评价模型

(1)建立

在我们设计的“印象因素层级图”中，“信贷风险”F 受“企业信誉”“企业实力”“企业供求稳定性”影响，分别记作 F_1,F_2,F_3，与前文的 AHP 层次分析法不同，这里的 F_1,F_2,F_3 它们彼此之间的相对重要性不容易确定，因此我们采用另一种多指标综合评价方法——熵权法来求 F。

熵权法实际上是对指标重要程度描述的一种方法，该方法以熵理论为核心。事件发生概率越高，相应会表现出更为突出的有序性，而有序程度越高，则其信息熵取值更大，相应的，其权重参数取值更小。基于此，在综合评价中如需应用多个指标，则可借助该方法完成指标赋权；此外，此方法是客观赋值法，即根据指标自身的数据分布特点即可求出结果。

Step1

构建一个 $n\times m$ 矩阵

$$X=(x_{ij})\times m(i=1,2,3...n;\ j=1,2,3...m)$$

表示个 n 企业的风险值矩阵。

Step2

将不同的指标分别归一化处理

$$b_{ij}=\frac{x_{ij}-\min\left(x_i\right)}{\max\left(x_i\right)-\min\left(x_i\right)}$$

得到新矩阵

Step3

计算各个指标的熵值

$$I_j=-k\sum_{i-1}^{m}p_{ij}\ln\left(p_{ij}\right),\quad (j=1,2,3\ldots\ldots m)$$

k通常可取$1/\ln(n)$

$$\text{其中}p_{ij}=\frac{b_{ij}}{\sum_{i=1}^{n}b_{ij}}\text{，若}b_{ij}<=0,\text{则}p_{ij}=\frac{1+b_{ij}}{\sum_{i=1}^{n}(1+b_{ij})}$$

Step4

求出指标权重

$$w_j=\frac{1-I_j}{\sum_{j=1}^{m}(1-I_j)}$$

Step5

求出综合评价的风险值

$$Y_i=\sum_{j=1}^{m}w_j\times p_{ij}$$

(2)数据处理

求解此模型所需要的三个指标 企业实力P,企业信誉C,企业稳定性S 在前面已经求出归一化的值，故此处无需处理

(3)求解

利用前文的熵权法步骤可求出三个指标各自的权重

熵值法计算权重结果汇总			
项	信息熵值 e	信息效用值 d	权重系数 w
credit 归一化	0.9934	0.0066	56.42%
稳定性	0.9979	0.0021	17.91%
power	0.9970	0.0030	25.67%

图表 10 熵权法权重

对P,C,S进行加权叠加，从而计算出风险值

$$F = w_1 P + w_2 C + w_3 S$$

5.1.5 银行信贷利率确定

(1)处理思路

根据附录3中的数据，我们分别可以对信誉等级为A，B，C的三类企业的贷款利率（R）和客户流失率（T）进行多项式拟合。可通过Numpy库得到n阶拟合函数

$$T = \alpha_0 + \alpha_1 R + \alpha_2 R^2 + ... + \alpha_n R^n$$

再通过计算下列银行放贷收益（M）公式，求出在贷款区间4%到15%中的最大收益求出A,B,C三个信贷评级的最佳贷款利率

$$M = R(1-T)Z$$

即 $M = R(1-\alpha_0 - \alpha_1 R - \alpha_2 R^2 - ... - \alpha_n R^n)Z$ -------公式（1）

对R求偏导，求出在4%到15%间的极小值点，若无极小值则直接取区间两端的点。

(2)数据处理

通过pandas库读入数据，将数据转成列表，R列表对应贷款利率，T列表对应客户流失率。

(3)求解

Step1

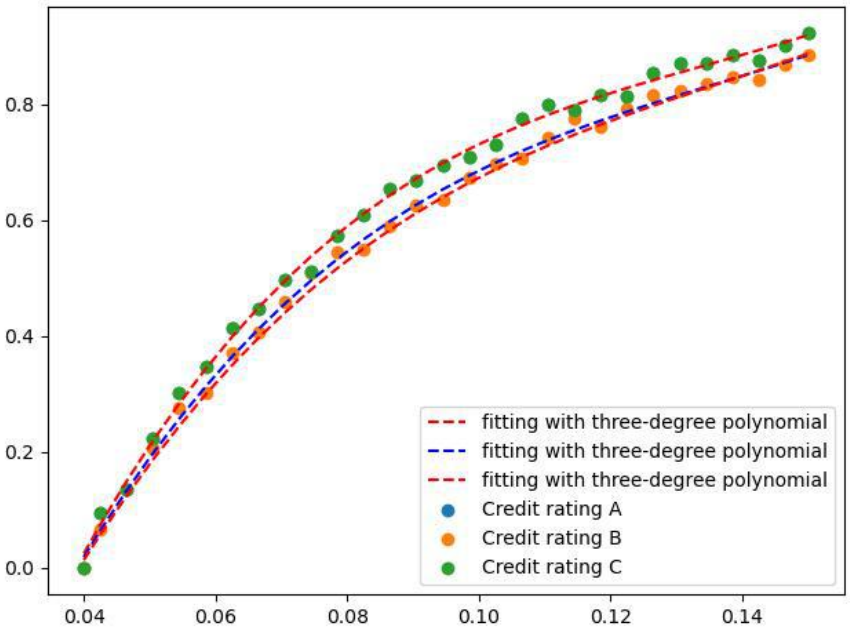
进行数据拟合，得出函数表达式

使用Numpy库中的.polyfit()函数对R列表和T列表进行拟合，最高位阶数为3。用matplotlib库进行绘图，绘制拟合后的函数图像，并打印输出A，B，C三个等级的函数表达式

$$T_A = 640.9R^3 - 258.6R^2 + 37.97R - 1.121$$ ---------公式（2）

$$T_B = 552.8R^3 - 225.1R^2 + 33.99R - 1.017$$ ---------公式（3）

$$T_C = 504.7R^3 - 207.4R^2 + 32.16R - 0.9735 \quad \text{---------公式（4）}$$



图表 11 三个等级的拟合后函数图像

Step2

使用 python 对上述公式（1）进行求解，将公式（2），（3），（4）代入公式（1），利用循环，取步长为 1，区间[0,10000]，归一化处理后就是步长为 0.0001，区间[0,1]，最后得出

信誉评级	最佳贷款利率	用户流失率
A	4.70%	3.95%
B	5.18%	4.06%
C	5.38%	4.12%

图表 12 不同信誉评级的最佳贷款利率

5.1.6 银行信贷策略模型

(1)建立

依照我们之前建立的“影响因素层级图”，企业风险值 F 和“信贷利率与客户流失量的关系” R 决定银行的信贷策略。

前文已经求出了最佳贷款利率，现在需要确定贷款额度。而贷款额度可以根据企业的风险值在总体的排名百分比来确定，即

贷款额度（ Z_{Ei} ）：

$$Z_{Ei}=\frac{F_i}{\sum_{i=1}^{n}F_i}Z \quad \text{----------公式（1）}$$

注意到题目要求银行的贷款额度必须在 $[10w,100w]$ 区间内，若上述函数求得的 Z_{Ei} 不在此区间内，则对其进行调整：

$$if(Z_{Ei}<10w):Z_{Ei}=0$$
$$else:Z_{Ei}=100$$

贷款利率根据信誉评级：

信誉评级	贷款利率
A	4.70%
B	5.18%
C	5.38%

(2)数据处理

将之前处理好的风险值 F 表格进一步处理，给出银行的放贷总额度，利用上述公式（1）求解即可得出银行给某个具体企业的贷款额度，加上前文已经确定的贷款利率，贷款策略 Y 即得解。

5.2 问题二

5.2.1 违约预测模型

(1)违约预测

依据附件 1 中的信息，不同信誉评级的企业中违约的企业的占比不同，以假设附件 1 中数据具有一般代表性，则可以通过违约占比刻画不同评级企业的违约概率。反之，可以用违约概率来预测信誉评级，再用上述信誉评价模型求出。

信誉评级	违约数	评级数	占比
A	0	27	0
B	1	38	1/38
C	2	34	1/17
D	24	24	1

(2)建立

我们需要根据附件 1 中的相关信息建立起违约预测的模型，考虑附件 1 中的各个信息：销进项负单占比，销进项作废发票占比，月均利润，月均利润率，月均订单数等为自变量；以违约量化为因变量，违约即为 1，无违约则为 0。

这正好可以使用二元 logit 回归模型对各指标进行相关性分析，求得回归系数和 p 值。

(3)结果分析

从上表 10 可知，将月均订单数 $n_{月}$，销项发票作废比 $r_{销废}$，进项负单比 $r_{进负}$，销项负单比 $r_{销负}$，进项发票作废比 $r_{进废}$，月均利润率 $r_{月}$，月均利润 $l_{月}$ 共 7 项为自变量，而将违约量化作为因变量 y 进行二元 Logit 回归分析，模型公式为：

$$\ln(p/(1-p)) = -0.511 - 0.125 * n_{月} + 5.587 * r_{销废} - 6.920 * r_{进负} - 2.405 * r_{销负} - 0.001 * r_{月} - 0.000 * l_{月} + 1.769 * r_{进废}$$

(其中 p 代表违约量化为 1 的概率，1- p 代表违约量化为 0 的概率)。

二元 Logit 回归分析结果汇总						
项	回归系数	标准误	z 值	p 值	OR 值	OR 值 95% CI
月均订单数	-0.125	0.044	-2.802	0.005	0.883	0.809 ~ 0.963
销项发票作废比	5.587	2.598	2.150	0.032	266.886	1.640 ~ 43427.559
进项负单比	-6.920	21.572	-0.321	0.748	0.001	0.000 ~ 2275419859979014.500
销项负单比	-2.405	5.429	-0.443	0.658	0.090	0.000 ~ 3770.933
月均利润率	-0.001	0.002	-0.609	0.543	0.999	0.995 ~ 1.003
月均利润	-0.000	0.000	-2.121	0.034	1.000	1.000 ~ 1.000
进项发票作废比	1.769	12.686	0.139	0.889	5.866	0.000 ~ 368690085895.586
截距	-0.511	0.649	-0.788	0.431	0.600	0.168 ~ 2.138
因变量: 违约量化						
McFadden R 方: 0.333						
Cox & Snell R 方：0.295						
Nagelkerke R 方：0.454						

图表 13 二元 logit 回归分析结果汇总

根据总结分析可知：销项发票作废比会对违约量化产生显著的正向影响关系，以及月均订单数，月均利润会对违约量化产生显著的负向影响关系。但是进项负单比，销项负单比，月均利润率，进项发票作废比并不会对违约量化产生影响关系。

二元 Logit 回归预测准确率汇总

		预测值		预测准确率	预测错误率
		0	1		
真实值	0	92	4	95.83%	4.17%
	1	14	13	48.15%	51.85%
汇总				85.37%	14.63%

图表 14 二元 logit 回归预测准确率汇总

通过模型预测准确率去判断模型拟合质量，从上表 11 可知：研究模型的整体预测准确率为 85.37%，模型拟合情况可以接受。当真实值为 0 时，预测准确率为 95.83%；另外当真实值为 1 时，预测准确率为 48.15%

(4)求解

通过建立模型，可以得到模型公式为：

ln(p/1-p)=-0.511-0.125*月均订单数 + 5.587*销项发票作废比-6.920*进项负单比-2.405*销项负单比-0.001*月均利润率-0.000*月均利润 + 1.769*进项发票作废比

即：

$$\ln\left(p/(1-p)\right)=\alpha_0+\alpha_1x_1+\alpha_2x_2+\alpha_3x_3+\alpha_4x_4+\alpha_5x_5+\alpha_6x_6$$

从而可求：

$$p=1-1/e^{(\alpha_0+\alpha_1x_1+\alpha_2x_2+\alpha_3x_3+\alpha_4x_4+\alpha_5x_5+\alpha_6x_6)}$$

p 即为违约概率，再根据图 9 中给出的违约数占比，可以评定该公司的信誉评级，进而套用上述企业信誉评价模型给出预测的企业信誉值 C 。

5.3 问题三

5.3.1 突发因素影响模型

(1)建立

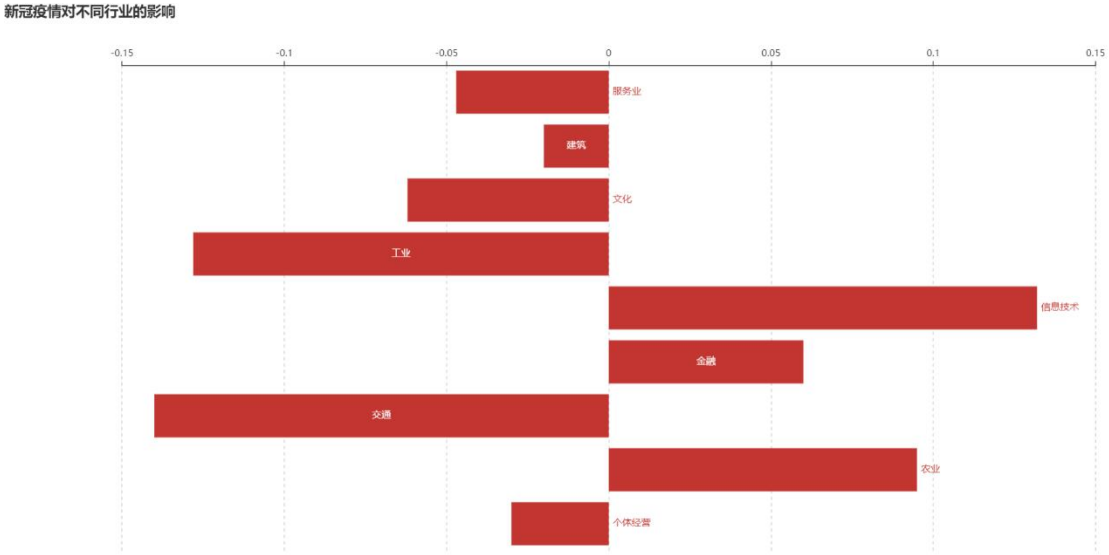
以“突发因素对行业的影响系数$\lambda_{行}$”近似估计“突发因素对企业的影响系数$\lambda_{企}$”，假设$\lambda_{行}$只会对企业实力P造成影响，不影响企业信誉C和企业稳定性S。

$$P_{新}=\lambda_{企}P=\lambda_{行}P$$

再将$P_{新}$代入问题二的模型即得解

(2)数据处理

我们在国家统计局官网中搜集新冠疫情期间的大量统计数据，得到对不同行业利润的增减影响情况δ

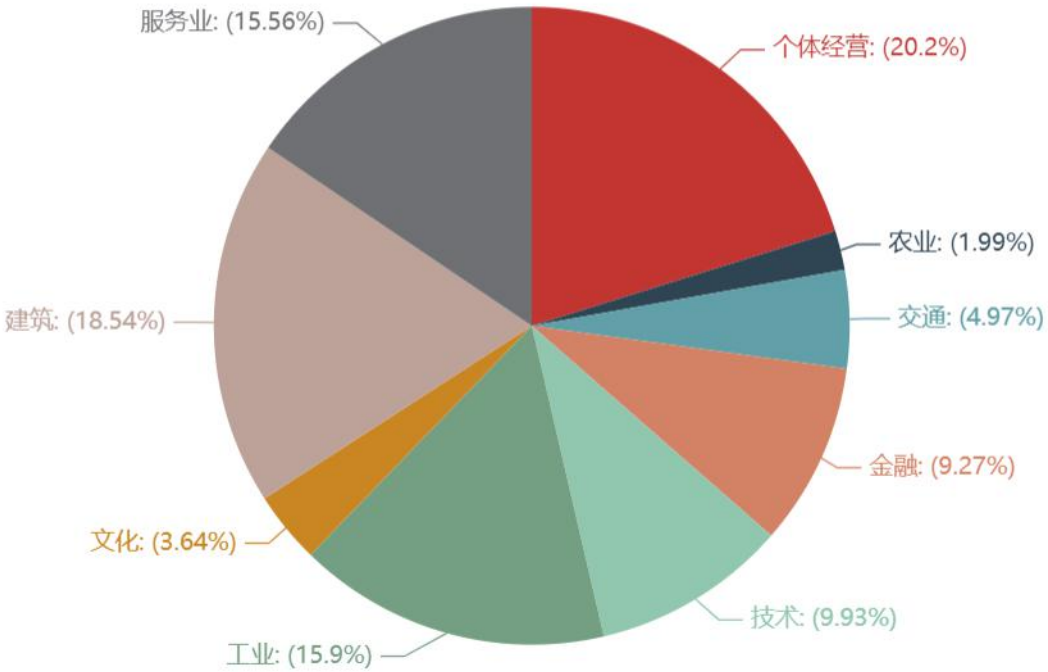


图表 15 新冠疫情对不同行业的影响

据此求出

$$\lambda_{行}=1+\delta$$

同时，对附件中的所有企业进行分类，划归到不同行业：



图表 16 附件二中各行业企业的占比

(3)求解

Step1

针对某个具体的企业，令

$$\lambda_{企}=\lambda_{行}$$

Step2

考虑突发因素对不同行业的影响，从而更新具体企业的实力值

$$P_{新}=\lambda_{企}P$$

Step3

将 $P_{新}$ 代入问题二的模型中求解即可得到企业的信贷风险数值。综合分析附件二中各企业的占比，在银行信贷对象企业中，个体经营所占的比重最大；但在新冠疫情影响下，个体经营出现衰退趋势。结合模型预测此时若对个体经营行业进行信贷业务，会有较大的违约风险，收益不如预期。

在疫情影响下，信息技术、农业、金融仍然实现了增长。在根据上述模型预测的信贷风险的基础下，对于这三个行业中信贷风险较小的企业优先贷款，并在信贷额度充足的情况下可适当增加额度。

六　模型的评价与推广

6.1 模型评价

优点：

（1）模型根据给出的数据进行详细具体的分析，考虑因素多，经过逐层建模，较为严谨，考虑维度全面，符合实际。

（2）模型的求解采用了 python 中的科学计算库 Numpy、Pandas；以及 Visio、Excel、Spssau 等专业工具，这些对于处理海量数据、绘制图表、拟合函数作用非常大，有助于模型求解的科学性、准确性以及高效性。

（3）实际应用需求强，可以为银行制定对小微企业的贷款策略提供参考。

缺点：

在建立数学模型时，部分条件考虑理想化，如：假设突发因素只影响企业实力，不影响企业信誉。

6.2 模型推广

通过分析中小微企业的历史交易记录和信贷记录等数据，可以预测银行信贷风险，也可以对无信贷记录的企业做评估，预测其信誉等级，从而为银行的信贷决策提供科学的依据。

七　参考文献

[1] 杨蓬勃,张成虎,张湘.基于Logistic回归分析的上市公司信贷违约概率预测模型研究[J].经济经纬,2009(02):144-148.

[2] 贾海涛,邱长溶.宏观因素对贷款企业违约率影响的实证分析[J].现代管理科学,2009(02):67-69+72.

[3] 高飞.基于 Logistic 回归模型的商业银行零售贷款客户违约分析——以MSLZ 银行为例[J].区域治理,2019(38):204-207.

[4] 孙雪梅.浅析商业银行中小企业信贷风险与管理[J].财经界（学术）,2010.

[5] 张莉.论中小企业贷款风险管理[J].南方金融,2014,(4).81-82,25.

[6] 姜启源，谢金星，叶俊.数学模型(第三版）[M].北京：高等教育出版社，2003.85-130.

[7] 戴国强,吴许均.基于违约概率和违约损失率的贷款定价研究[J].国际金融研究,2005(10):43-48.

[8] 国家统计局.季度数据.

https://data.stats.gov.cn/easyquery.htm?cn=B01

[9] 高英娟.熵权—功效系数法财务风险预警在生物制药行业的应用研究[D].云南大学,2018.

[10] SPSSAU 数据科学在线网站——AHP 层次分析法

https://spssau.com/front/spssau/helps/weights/ahp.html

八　附录

程序编号	T1-1	文件名称	delete_D.py
说明	对信息进行筛选，剔除信誉 D 级的企业		

```
# 剔除信誉 D 级企业
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '销项发票信息', usecols=(0, 7))
D = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '企业信息', usecols=(0, 2))

delete = []
for i in range(123):
    if D.iloc[i, 1] == 'D':
        delete.append(i)
D = D.drop(labels=delete, axis=0)
print(D)
D.set_index('企业代号', inplace=True)
data_1 = data.value_counts()

list_1 = []
list_2 = []
for i in D.index.to_list():
    print(i)
    list_1.append(i)
    E = data_1.loc[i]
    print(E)
    if len(E) > 1:
        list_2.append(E.iloc[1] / (E.iloc[0] + E.iloc[1]))
    else:
        list_2.append(0 / E.iloc[0])

S1 = pd.Series(list_2, index=list_1)
print(S1)
S1.to_excel('C:\\Users\\10049\\Desktop\\2.xlsx')
```

程序编	T1-2	文件名称	select_1.py

号			
说明	计算供求稳定性，计算利润		

```
# 供求稳定、筛选、利润
import numpy as np
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\f2.xlsx', '
进项发票信息', usecols=(0, 2, 6, 7))
sale = pd.read_excel('C:\\Users\\10049\\Desktop\\f2.xlsx', '
销项发票信息', usecols=(0, 2, 6, 7))
D = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '企业信
息', usecols=(0, 2))
# 剔除 D 评级企业
delete = []
for i in range(123):
    if D.iloc[i, 1] == 'D':
        delete.append(D.iloc[i, 0])

D.set_index('企业代号', inplace=True)
D = D.drop(labels=delete, axis=0)

data.set_index('发票状态', inplace=True)
data_2 = data.drop(labels='作废发票', axis=0)
data_2.set_index('企业代号', inplace=True)
data_1 = data_2.drop(labels=delete, axis=0)

# data_1.to_excel('C:\\Users\\10049\\Desktop\\2.xlsx')

sale.set_index('发票状态', inplace=True)
sale_2 = sale.drop(labels='作废发票', axis=0)
sale_2.set_index('企业代号', inplace=True)
sale_1 = sale_2.drop(labels=delete, axis=0)

# sale_1.to_excel('C:\\Users\\10049\\Desktop\\3.xlsx')

l1 = []
l2 = []
l3 = []
ser = pd.Series([' '])
for i in D.index.to_list():
    a = data_1.loc[i]
    b = sale_1.loc[i]
```

```
    if type(a) == type(ser):
        a = a.to_frame()
        a = pd.DataFrame(a.values.T, columns=a.index)

    if type(b) == type(ser):
        b = b.to_frame()
        b = pd.DataFrame(b.values.T, columns=b.index)

    a_g = a.groupby('开票日期').sum()
    b_g = b.groupby('开票日期').sum()

    l_a = a_g.index.to_list()
    l_b = b_g.index.to_list()
    time_1 = min(l_a[0], l_b[0])
    time_2 = max(l_a[-1], l_b[-1])
    pda = pd.date_range(start=time_1, end=time_2, freq='M')
    # pd = pd.date_range(start=time_1, end=time_2)
    a_t = a_g.reindex(pda, fill_value=0)
    b_t = b_g.reindex(pda, fill_value=0)
    #
    for j in pda:
        j = str(j)[0:7]
        l1.append(i)
        l2.append(j)
        print(i, ' ', j)
        l3.append((b_t.loc[j].sum() -
a_t.loc[j].sum()).iloc[0])
        print((b_t.loc[j].sum() - a_t.loc[j].sum()).iloc[0])

data_out = pd.DataFrame({
    '企业代号': l1,
    '开票时间（月）': l2,
    '销进差': l3
})
print(data_out)
data_out.describe()

# data_out.to_excel('C:\\Users\\10049\\Desktop\\盈利趋
势.xlsx')

l1 = []
l2 = []
```

```
l3 = []
data_out.set_index('企业代号', inplace=True)
for i in data_out.index.unique():
    a = data_out.loc[i, '销进差']
    b = a.describe()
    l1.append(i)
    l2.append(a.std())

for i in l2:
    l3.append((i-np.min(l2))/(np.max(l2)-np.min(l2)))

data_o = pd.DataFrame({
    '企业代号': l1,
    '标准差': l2,
    '标志差（归一化）': l3
})

print(data_o)
data_o.to_excel('C:\\Users\\10049\\Desktop\\4.xlsx',
sheet_name='Sheet2')
```

程序编号	T1-3	文件名称	norm.py
说明	进行数据归一化		

```
# 归一化、最终策略
import numpy as np
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\建模文件
\\risk2.xlsx', 'Sheet1')

# 数据归一化处理
min = data['credit'].min()
max = data['credit'].max()

l1 = []
for i in data['credit']:
    l1.append((i - min) / (max - min))
```

```
ser = pd.Series(l1)

data.loc[:, 'credit 归一化'] = ser

data.set_index('企业代号', inplace=True)

print(data)
data.to_excel('C:\\Users\\10049\\Desktop\\建模文件
\\risk2.xlsx', 'Sheet1')
```

程序编号	T1-4	文件名称	power_compute.py
说明	计算企业实力相关的月均订单数，月均利润，月均利润率		

```
# 利润、利润率、订单数
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '进
项发票信息', usecols=(0, 2, 6, 7))
sale = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '销
项发票信息', usecols=(0, 2, 6, 7))
D = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '企业信
息', usecols=(0, 2))
# 剔除 D 评级企业
delete = []
for i in range(123):
    if D.iloc[i, 1] == 'D':
        delete.append(D.iloc[i, 0])

D.set_index('企业代号', inplace=True)
D = D.drop(labels=delete, axis=0)

data.set_index('发票状态', inplace=True)
data_2 = data.drop(labels='作废发票', axis=0)
data_2.set_index('企业代号', inplace=True)
data_1 = data_2.drop(labels=delete, axis=0)

sale.set_index('发票状态', inplace=True)
sale_2 = sale.drop(labels='作废发票', axis=0)
sale_2.set_index('企业代号', inplace=True)
sale_1 = sale_2.drop(labels=delete, axis=0)
```

```
average_list = []
average_sale = []
average_saleRa = []
average_cov = []
name = []

for i in D.index.to_list():

    a = data_1.loc[i]
    b = sale_1.loc[i]

    if type(a) == pd.Series:
        a = a.to_frame()
        a = pd.DataFrame(a.values.T, columns=a.index)

    if type(b) == pd.Series:
        b = b.to_frame()
        b = pd.DataFrame(b.values.T, columns=b.index)

    a_g = a.groupby('开票日期').sum()
    b_g = b.groupby('开票日期').sum()

    l_a = a_g.index.to_list()
    l_b = b_g.index.to_list()
    time_1 = min(l_a[0], l_b[0])
    time_2 = max(l_a[-1], l_b[-1])
    time_3 = (time_2 - time_1).days / 30

    name.append(i)
    average_list.append(len(b) / time_3)

    sale_money = b_g.sum() - a_g.sum()


    average_sale.append(sale_money.iloc[0])

    average_saleRa.append(sale_money.iloc[0] /
a_g.sum().iloc[0])


data_out = pd.DataFrame({
    '企业代号': name,
```

```
    '月均订单数': average_list,
    '月均利润': average_sale,
    '月均利润率': average_saleRa,
})

# print(data_out)
# print(data_out['月均订单数'].corr(b['月均利润']))
data_out.to_excel('C:\\Users\\10049\\Desktop\\5.xlsx')
```

程序编号	T1-5	文件名称	function_fit.py
说明	多项式拟合		

```
# 多项式拟合
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\f3.xlsx',
sheet_name='Sheet1')
# print(data)
# print('协方差:\n', data.cov(), '\n')
# print('相关系数:\n', data.corr(), '\n')
x = data.loc[1:, '贷款年利率'].to_list()
y = data.loc[1:, '客户流失率'].to_list()

# y1 = data.loc[1:, 'Unnamed: 2'].to_list()
# y2 = data.loc[1:, 'Unnamed: 3'].to_list()

a = np.polyfit(x, y, 3)
# a1 = np.polyfit(x, y1, 3)
# a2 = np.polyfit(x, y2, 3)
b = np.poly1d(a)
# b1 = np.poly1d(a1)
# b2 = np.poly1d(a2)
c = b(x)
# c1 = b1(x)
# c2 = b2(x)
plt.scatter(x, y, marker='o', label='Credit rating A')
plt.plot(x, c, ls='--', c='blue', label='fitting with
three-degree polynomial')
plt.legend()
```

```
# plt.scatter(x, y1, marker='o', label='Credit rating B')
# plt.plot(x, c1, ls='--', c='red', label='fitting with
three-degree polynomial')
# plt.legend()
#
# plt.scatter(x, y, marker='o', label='Credit rating C')
# plt.plot(x, c2, ls='--', c='green', label='fitting with
three-degree polynomial')
# plt.legend()
#
# plt.show()

print('red: ', b, '\n')
# print('blue: ', b1, '\n')
# print('green: ', b2, '\n')
```

程序编号	T1-6	文件名称	strategy.py
说明	计算供求稳定性		

```
# 最终策略
import numpy as np
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\建模文件
\\risk2.xlsx', 'Sheet1')
# 最终贷款策略计算
data.set_index('企业代号', inplace=True)

max_xa = 0.047

max_xb = 0.0518

max_xc = 0.0538

for i in data.index:
    cred = data.loc[i, 'risk']
    s = data['risk'].sum()
    if data.loc[i, '信誉评级'] == 'A':
        data.loc[i, '信赖程度'] = (1 - cred)/s
        data.loc[i, '贷款利率'] = max_xa
```

```
    if data.loc[i, '信誉评级'] == 'B':
        data.loc[i, '信赖程度'] = (1 - cred)/s
        data.loc[i, '贷款利率'] = max_xb
    if data.loc[i, '信誉评级'] == 'C':
        data.loc[i, '信赖程度'] = (1 - cred)/s
        data.loc[i, '贷款利率'] = max_xc
    if data.loc[i, '信誉评级'] == 'D':
        data.loc[i, '信赖程度'] = 0


print(data.loc[:, '信赖程度'])

print(data)
data.to_excel('C:\\Users\\10049\\Desktop\\建模文件
\\risk2.xlsx', 'Sheet1')
```

程序编号	T1-7	文件名称	compute_2.py
说明	计算作废比		

```
# 作废比
import numpy as np
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '进
项发票信息', usecols=(0, 2, 6, 7))
sale = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '销
项发票信息', usecols=(0, 2, 6, 7))

data.set_index('企业代号', inplace=True)
sale.set_index('企业代号', inplace=True)


def search_t(data_f, l1, l2):
    for i in data_f.index.unique():
        l1.append(i)
        data_1 = data_f.loc[i]
        b = 0
        for j in data_1['发票状态']:
            if '作废发票' == j:
                b += 1
        l2.append(b / len(data_1))
```

```
l_1 = []
l_2 = []
search_t(data, l_1, l_2)
print(l_1)
print(l_2)

data_out = pd.DataFrame({
    '企业': l_1,
    '进项发票作废比': l_2,
})

print(data_out)

l_3 = []
search_t(sale, l_1, l_3)
ser = pd.Series(l_3)
data_out['销项发票作废比'] = ser

print(data_out)
data_out.to_excel('C:\\Users\\10049\\Desktop\\建模文件\\f1 作废比.xlsx')
```

程序编号	T2-1	文件名称	Select_2.py
说明	计算供求稳定性（含信誉 D 级企业）		

```
# 筛选、稳定性、标准差
import numpy as np
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\f2.xlsx', '进项发票信息', usecols=(0, 2, 6, 7))
sale = pd.read_excel('C:\\Users\\10049\\Desktop\\f2.xlsx', '销项发票信息', usecols=(0, 2, 6, 7))

data.set_index('发票状态', inplace=True)
data_1 = data.drop(labels='作废发票', axis=0)
data_1.set_index('企业代号', inplace=True)

sale.set_index('发票状态', inplace=True)
```

```
sale_1 = sale.drop(labels='作废发票', axis=0)
sale_1.set_index('企业代号', inplace=True)

l1 = []
l2 = []
l3 = []
ser = pd.Series([' '])
for i in data_1.index.unique():
    a = data_1.loc[i]
    b = sale_1.loc[i]

    if type(a) == type(ser):
        a = a.to_frame()
        a = pd.DataFrame(a.values.T, columns=a.index)

    if type(b) == type(ser):
        b = b.to_frame()
        b = pd.DataFrame(b.values.T, columns=b.index)

    a_g = a.groupby('开票日期').sum()
    b_g = b.groupby('开票日期').sum()

    l_a = a_g.index.to_list()
    l_b = b_g.index.to_list()
    time_1 = min(l_a[0], l_b[0])
    time_2 = max(l_a[-1], l_b[-1])
    pda = pd.date_range(start=time_1, end=time_2, freq='M')
    # pd = pd.date_range(start=time_1, end=time_2)
    a_t = a_g.reindex(pda, fill_value=0)
    b_t = b_g.reindex(pda, fill_value=0)
    #
    for j in pda:
        j = str(j)[0:7]
        l1.append(i)
        l2.append(j)
        # print(i, ' ', j)
        l3.append((b_t.loc[j].sum() -
a_t.loc[j].sum()).iloc[0])
        # print((b_t.loc[j].sum() - a_t.loc[j].sum()).iloc[0])

data_out = pd.DataFrame({
    '企业代号': l1,
    '开票时间（月）': l2,
```

```
    '销进差': l3
})
# print(data_out)

l1 = []
l2 = []
l3 = []
data_out.set_index('企业代号', inplace=True)
for i in data_out.index.unique():
    a = data_out.loc[i, '销进差']
    b = a.describe()
    l1.append(i)
    l2.append(a.std())

for i in l2:
    l3.append((i-np.min(l2))/(np.max(l2)-np.min(l2)))

data_o = pd.DataFrame({
    '企业代号': l1,
    '标准差': l2,
    '标志差（归一化）': l3
})

print(data_o)
data_o.to_excel('C:\\Users\\10049\\Desktop\\f2 供应稳定.xlsx',
sheet_name='Sheet2')
```

程序编号	T2-2	文件名称	compute_3.py
说明	计算负单比		

```
# 负单比
import numpy as np
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '进
项发票信息', usecols=(0, 2, 6, 7))
sale = pd.read_excel('C:\\Users\\10049\\Desktop\\1.xlsx', '销
项发票信息', usecols=(0, 2, 6, 7))

data.set_index('发票状态', inplace=True)
data = data.drop(labels='作废发票', axis=0)
```

```
sale.set_index('发票状态', inplace=True)
if '作废发票' in sale.index.to_list():
    sale = sale.drop(labels='作废发票', axis=0)

data.set_index('企业代号', inplace=True)
sale.set_index('企业代号', inplace=True)


def search_t(data_f, l1, l2):
    for i in data_f.index.unique():
        l1.append(i)
        data_1 = data_f.loc[i]
        b = 0
        print(data_1)
        if type(data_1) == pd.Series:
            if data_1.loc['价税合计'] < 0:
                l2.append(1)
            else:
                l2.append(0)
        else:
            for j in data_1.loc[:, '价税合计']:
                if j < 0:
                    b += 1
            l2.append(b / len(data_1))


l_1 = []
l_2 = []
search_t(data, l_1, l_2)
print(l_1)
print(l_2)

data_out = pd.DataFrame({
    '企业代号': l_1,
    '进项负单比': l_2,
})

print(data_out)

l_3 = []
search_t(sale, l_1, l_3)
ser = pd.Series(l_3)
data_out['销项负单比'] = ser
```

```
print(data_out)
data_out.to_excel('C:\\Users\\10049\\Desktop\\建模文件\\f1 负单比.xlsx')
```

程序编号	T2-3	文件名称	Predict_credict.py
说明	违规预测与信誉预测		

```
# 违规预测
import pandas as pd
import math

data = pd.read_excel('C:\\Users\\10049\\Desktop\\建模文件\\预测违约率.xlsx')

input_1 = -6.920
output_1 = -2.405
average_ra = -0.001
average_mon = -0.000
average_number = -0.125
input_2 = 1.769
output_2 = 5.587

data.loc[:, '违约预测'] = input_1 * data['进项负单比'] + output_1 * data['销项负单比'] + average_ra * data['月均利润率'] + \
                      average_number * data['月均订单数'] + input_2 * data['进项发票作废比'] + output_2 * data['销项发票作废比'] - 0.511

# average_num = -0.127
# out_put = 5.902
#
# data.loc[:, '违约预测'] = average_num * data['月均订单数'] + out_put * data['销项发票作废比'] - 0.808
l1 = []
l2 = []
for i in data['违约预测']:
    p = math.exp(i) / (1 + math.exp(i))
    l1.append(p)
    if p > 1 / 17:
        l2.append("D")
```

```
    elif p > 1 / 38:
        l2.append('C')
    elif p > 1 / 10000:
        l2.append('B')
    else:
        l2.append('A')

data.loc[:, '违规概率'] = pd.Series(l1)
data.loc[:, '预测信誉评级'] = pd.Series(l2)
print(data)

data.to_excel('C:\\Users\\10049\\Desktop\\建模文件\\f1 预测违约.xlsx')
```

程序编号	T3-1	文件名称	power3_compute.py
说明	换算利润		

```
# 换算利润
import pandas as pd
import numpy as np

data = pd.read_excel('C:\\Users\\10049\\Desktop\\建模文件\\power_f2.xlsx')
data_1 = pd.read_excel('C:\\Users\\10049\\Desktop\\建模文件\\新冠对行业的影响.xlsx')

data.set_index('企业代号', inplace=True)
data_1.set_index('行业类别')


def fun (b):
    data.loc[i, '月均订单数'] = data.loc[i, '月均订单数'] * b
    data.loc[i, '月均利润'] = data.loc[i, '月均利润'] * b


for i in data.index:
    a = data.loc[i, '行业类别']
    if a in data_1.index:
        fun(data_1.loc[a, '最终系数'])

data.to_excel('C:\\Users\\10049\\Desktop\\建模文件
```

```
\\power_f3.xlsx')
```

程序编号	T3-2	文件名称	check.py
说明	检查贷款策略分配的额度		

```
# 贷款额度检查
import pandas as pd

data = pd.read_excel('C:\\Users\\10049\\Desktop\\建模文件\\第二问最终结果.xlsx')

data['贷款额度(万元)'] = data['信赖程度'] * 10000
if data['贷款额度(万元)'].max() > 100:
    print('存在贷款额度超额情况')
if data['贷款额度(万元)'].min() < 10:
    print('存在贷款额度不足')
data.to_excel('C:\\Users\\10049\\Desktop\\建模文件\\第二问最终结果.xlsx')
```