中小微企业的信贷决策研究

摘要

本文针对银行放贷问题，建立了模糊评价和 TOPSIS 评价模型量化企业信贷风险的量化问题；建立了基于目标优化的双阶段信贷策略选择模型，决策出在银行最优信贷策略；建立了基于动态蛛网理论的突发情况信贷调整模型，协助银行在企业经历突发状况时的信贷策略调整。

针对问题一，首先，数据预处理，发现数据存在“作废发票”，两级分化明显等问题；进行数据规格化，构建企业信贷风险影响指标；利用数据可视化，探查指标与企业信誉评级的影响程度与选择该指标的必要性。**然后**，根据预处理的结果，构建关于企业信誉指数的三级指标体系，采用模糊评价和 TOPSIS 评价分别从定性和定量两个方面进行企业信贷风险量化。**最后**，建立基于目标优化的双阶段信贷策略选择模型，阶段一建立单目标优化模型，以贷款金额作为决策变量，以风险最小作为目标函数；阶段二通过指数函数拟合贷款利率与客户流失率关系，建立银行期望收益最大化模型，求得信贷策略最优解。**信贷风险量化结果**：ABCD 级企业的信贷风险指数值约为 0.9，0.3，0.1，<<0.1，指数值越高即表明企业信贷风险越低；**信贷策略选择结果**：银行对于 ABC 级企业的贷款额度以 10:4:1 的比例进行分配可以达到信贷风险指数值最大，安全性最高，最优年利率设置为 5.8%、6.2%和 6.3%，银行的期望收益最大，对 D 级企业不予放贷。

针对问题二，首先，利用熵权法计算问题一中建立的三级指标体系的决策权重值；**接着**，利用建立的 TOPSIS 评价体系，计算所有企业信贷风险的定量量化结果，即信贷风险评估指数；**然后**，利用模糊评价指标区间，得到企业信贷风险的定性量化结果，即信誉级别。**最后，**对于信贷策略的选择，利用问题一建立的基于目标优化的双阶段信贷策略选择模型，求解最优贷款额度分配和贷款利率选择。**风险量化结果显示**：ABCD 级企业各有 103 家，127 家，47 家，20 家。同时，利用 LightGBM 算法进行验证，结果显示两者训练结果有 99.5%的匹配度，表明模型的准确性和适用性。**信贷策略结果显示**：对于 1 亿元总额，对所有 A 级企业和 79.8%的 B 级企业放贷 100w 元，对所有 C 级企业及 21.2%的 B 级企业放贷 10w 元可以达到信贷风险指数值最大，此时，ABC 级企业的最优年利率为 5.8%、6.2%和 6.3%。**最后，**为了验证了结果的可靠性，建立了多因素方差分析模型，所有因素均通过了显著性检验。

针对问题三，首先，建立了基于动态蛛网理论的突发情况信贷调整模型，引入突发情况因子，包括人均可支配收入和突发情况导致的额外成本，模拟了突发情况发生后放贷方与贷款方之间的动态博弈过程。**接着，**通过对于信贷产品金额收敛性分析，发现突发情况对于贷款方资金成本的影响，从而调整额外投入成本值稳定市场。**然后，**利用最大似然估计法对动态蛛网模型进行回归拟合，得到供求关系弹性曲线，确定蛛网模型的收敛性。**最后，**将结果带入价格收敛条件，得出调整之后的信贷策略。调整之后的信贷策略结果如图 7.2 所示。

关键词：数据预处理、TOPSIS 评价模型、单目标规划、熵权法、多因素方差分析动态蛛网理论、最大似然估计

1、问题背景与重述

1.1 问题背景

商业银行作为高风险行业，以安全性和稳健性作为发展战略，制定正确的信贷政策能够使银行的信贷管理保持理想的水平，避免过大风险[1]。近年来，中小微企业快速发展，逐渐成为中国经济发展的主要推动力，但是由于其规模相对较小，缺少抵押资产，所以银行会综合评估企业的资金实力、供求关系、信誉情况等，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，并对信誉高、信贷风险小的企业给予利率优惠。

1.2 问题要求

为了在保证银行利益的前提下，制定最合理的信贷政策，银行需要首先根据中小微企业的实力、信誉对其信贷风险做出评估，然后依据各种因素确定是否放贷及贷款额度、利率和期限等信贷策略。

现在已知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。需要我们根据实际和所提供的的数据信息，通过建立数学模型研究对中小微企业的信贷政策，主要解决以下问题：

（1） 对附件 1 中 123 家有信贷记录的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 家无信贷记录的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。

2、问题分析

2.1 问题一的分析

针对问题一信贷风险量化和信贷策略选择，首先，需要进行数据预处理，分析附件数据，发现存在“作废发票”，正负值价税总额，两级分化现象明显等问题。针对问题进行数据规格化，构建企业月度进销项价税总额及其增长率、退款价税总额及其增长率、大单交易额和企业税率值等企业信贷风险影响指标。同时，利用数据可视化，探查指标与企业信誉评级的影响程度与选择该指标的必要性。接着，根据预处理的结果，构建关于企业信誉指数的三级指标体系，采用模糊评价和 TOPSIS 评价分别从定性和定量两个方面进行企业信贷风险量化。最后，建立基于目标优化的双阶段信贷策略选择模型，阶段一以贷款金额作为决策变量，以风险最小作为目标函数，构造单目标优化模型；阶段二通过指数函数拟合贷款利率与客户流失率函数，建立银行期望收益模型，计算当期望收益最大时企业信贷年利率，即为最优解。

2.2 问题二的分析

针对问题二对于无信贷记录的企业信贷风险的量化和信贷策略的选择，首先利用熵权法计算问题一中建立的三级指标体系的决策权重值；接着利用建立的TOPSIS 评价体系，计算得到所有企业信贷风险的定量量化结果，即信贷风险评估指数；然后，利用建立的模糊评价指标，得到企业信贷风险的定性量化结果，即信誉级别。对于信贷策略的选择，由于此时的场景即为问题一的具体化，因此直接利用问题一建立的基于目标优化的双阶段信贷策略选择模型，求解最优贷款额度分配和贷款利率选择。

2.3 问题三的分析

针对问题三企业经历突发状况时银行的信贷策略调整，不同行业、不同类别企业所受的影响也各不相同。突发事件一方面通过对企业的现金流、生产经营活动等方面产生直接影响，一方面通过对整体行业、银行信贷政策等方面产生间接影响。为了能够有效的模拟突发情况发生后放贷方与贷款方之间的动态博弈过程，建立基于动态蛛网理论的突发情况信贷调整模型，引入突发情况因子，包括人均可支配收入和突发情况导致的额外成本。接着，通过对于信贷产品价格收敛性分析，发现突发情况对于贷款方资金成本的影响，从而调整相应额外投入成本值。然后，利用最大似然估计法对动态蛛网模型进行回归拟合，得到供求关系弹性曲线，确定蛛网模型的收敛性，最后，将结果带入价格收敛条件，得出调整之后的信贷策略。

3、模型假设

1、假设企业的信贷风险仅受附件中所给数据的影响。
2、假设相关发票数据准确，能够真实反映企业相关经营状况。
3、假设“作废发票”的相关数据不会对企业的收益和信誉产生影响。
4、假设企业都尽可能申请最大贷款。

4、符号说明

符号	表示含义
in_total_i	第 i 家企业的进项价税总额
in_rate	进项价税总额增长率
out_total_i	第 i 家企业的销项价税总额
$back_i$	第 i 家企业的退款价税总额
$back\_rate$	退款价税总额增长率
$large\_deals_j$	第 j 月的大单交易额
tax_ave_i	第 i 家企业的缴税率均值
$D_{m\times n}$	n 个指标 m 个企业的决策矩阵
C	企业的信贷风险评估指数

p_i	银行对于信誉级别为 i 企业期望收益
E_i	信息熵
w_i	第 i 项指标的决策权重

5、问题一模型的建立与求解

5.1 数据预处理

在对问题进行分析之前，首先需要进行数据预处理，宏观把控数据情况。将附件数据进行整理分析，发现存在“作废发票”，正负值价税总额，两级分化现象明显等问题。因此，本文对相关数据进行如下预处理。

5.1.1 数据规格化确立

附件一中，收集的是 123 家企业在 2017 年 7 月 18 日到 2019 年 12 月 18 日的所有进销项价税数据信息。首先，对于“作废发票”数据，由于其对企业的收益和信誉不会产生影响，所以直接进行删除处理；接着，为了能够体现数据变化情况，建立基于时间序列的风险评估模型，需要采用如下计算进行数据预处理：

（1）企业月度进销项价税总额及其增长率

以月为时间单位，分别计算所有企业的月度进销项价税总额及其增长率。针对第 i 家企业，其中：

进项价税总额为 $in\_total_i = \sum_j in_{ij}$ ，增长率为：

$$in\_rate = in\_total_{i+1} \Big/ in\_total_i$$

销项价税总额为 $out\_total_i = \sum_j out_{ij}$ ，增长率为：

$$out\_rate = out\_total_{i+1} \Big/ out\_total_i$$

此时，计算的总额不包括“负数发票”数据。

（2）退款价税总额及其增长率

根据题意可知，“负数发票”数据指购方因故发生退货并退款，即购方不满意企业提供服务，对其信誉等级会产生负面影响，因此对于第 i 家企业进行如下计算：

退款价税总额为 $back_i = \sum_j negative_{ij}$ ，增长率为：

$$back\_rate = back_{i+1} \Big/ back_i \quad ;$$

（3）大单交易额确定

从附件 1 分析得，同一企业中存在销售金额两级分化严重的情况，导致数据处理的非针对性，且较小值容易被忽略，所以需要确立交易阈值 k，根据所有企业的金额变化情况，确定统一阈值。超过阈值部分，规定为大单交易额

$large\_deals_j$。

（4）企业税率值

不同企业由于不同的信誉级别和不同的资金实力，其纳税比率存在差异，这些差异也直接影响了银行对于企业的信誉评估。因此，计算 n 个企业平均税率值为：

$$tax\_ave_i = \frac{\sum_j tex_{ij} / money_{ij}}{n}$$

5.1.2 数据可视化分析

对于步骤一中确定的规格化数据，进行数据可视化，探查数据的时间和空间规律，从而确定企业信誉指数的影响因素。探查结果如下：

（1）信誉评级四级企业的进项和销项价税总额对比

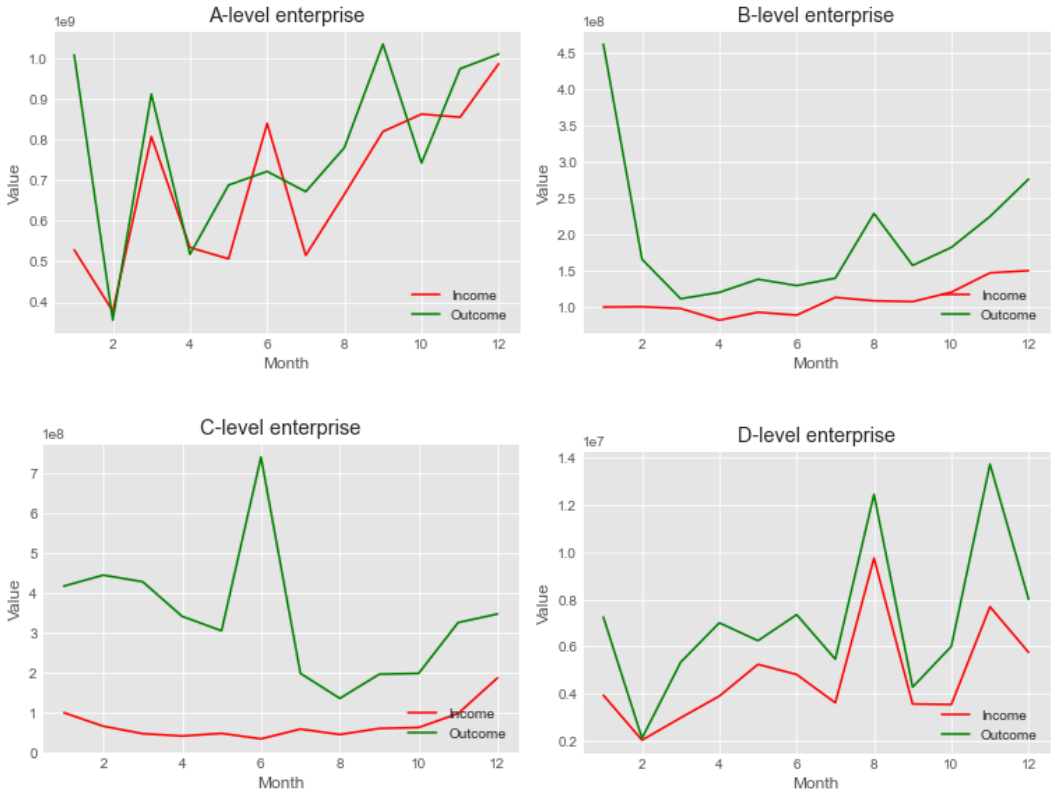

图 5-1 四级企业的进项和销项价税总额折线对比图

分析图 5-1 可知，信誉评级与企业的进项和销项价税总额有极大的相关性。信誉评级越高的企业，一方面进销商品价值总额越大，另一方面供需的需求正相关性也越大。因此，本方案利用进项和销项总金额和总成交笔数的差值作为企业信誉评级影响因素。

（2）大单交易阈值确定

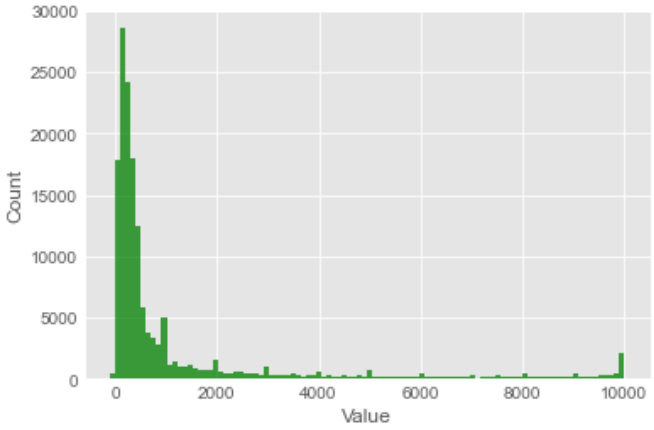

图 5-2 企业交易金额分布

图 5-2 为企业交易金额的数据量统计分布图，分析图可知，企业交易金额存在很严重的两级分化情况，单笔交易金额一般位于区间（0,4000）之内，少量位于（4000,10000）之间，考虑数据的稀疏性，我们选取 10000 作为大单的阈值 k，由此即可精确反应企业的资金实力情况。

（3）四级企业交易成功数和退货数对比

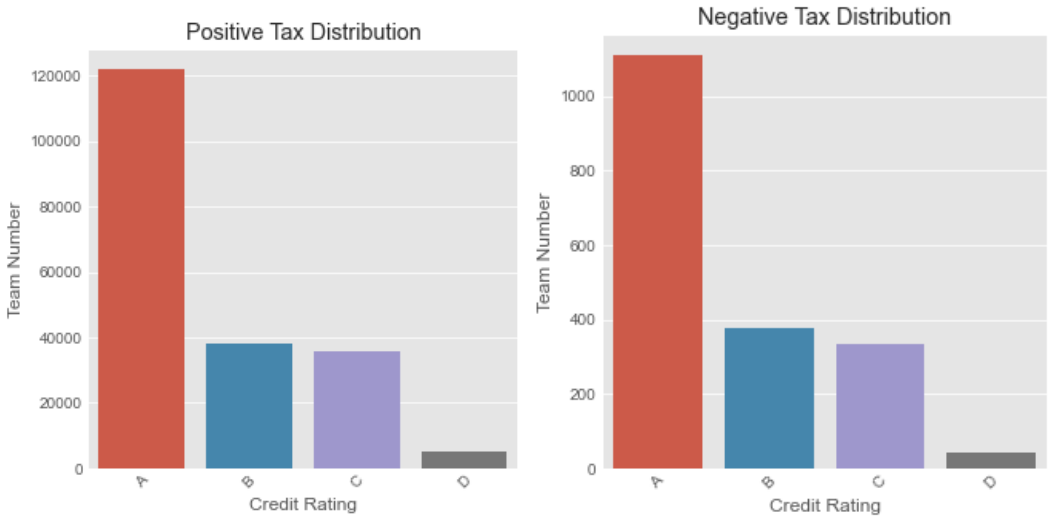

图 5-3 四级企业正价税和负价税数量的柱状图

图 5-3 为信誉评级 ABCD 四级的企业交易成功笔数和退货笔数的分布对比图，可以发现，A 级企业的交易成功率为 99.05%，B 级企业的交易成功率为 95.32%，C 级企业的交易成功率为 89.11%，D 级企业的交易成功率为 83.33%，同时 A 级企业交易成功笔数远远大于其他企业，因此企业的信达风险与其交易成功率有极大的相关性。

5.2 企业信贷风向指标体系建立

根据数据预处理结果，我们可以分析出企业的信誉指数与资金实力、供求关系稳定性和信贷风险有极大的相关性，资金实力越雄厚，供求关系越稳定，信贷风险越低，企业的信誉级别越高。因此，本文构建关于企业信誉指数的指标体系如表 5-1 所示。

表 5-1 指标体系表

一级指标	二级指标	影响因素（月度）
企业的信誉等级	资金实力	入账总额
		大单交易额
		大单交易额增长率
	供求关系	进项总笔数
		销项总笔数
		进销规模比值
	信贷风险	价税总额增长率
		退款总额增长率
		企业税率平均值

其中，入账总额=销项总额-进项总额，进销规模比值=进项总笔数/销项总笔数。

5.3 企业信贷风险量化结果

本方案采用模糊评价和 TOPSIS 评价分别从定性和定量[2]两个方面进行企业信贷风险量化。通过模糊评价，得出企业信誉级别在不同等级下，九项影响指标的取值范围，从而可以定性的解决风险分级问题；通过 TOPSIS 评价，建立企业信贷风险指数，定量量化风险程度，且为企业做信贷决策提供支持。

5.3.1 信誉级别模糊评价

本文采用综合评价模型量化信贷风险级别，选取附件 1 中数据的九项指标取样，根据所有取样值的区间分布，将区间进行合理划分，得到各等级的对应指标值，如下表所示：

表 5-2 评价标准

时间单位：月	A	B	C	D
入账总额	>6000w	4000w-6000w	1000w-4000w	<1000w
大单交易额	>1000w	500w-1000w	10w-500w	<10w
大单交易额增长率	>0.5	0.3-0.5	0-0.3	<0
进项总笔数	>100	60-100	40-60	<40
销项总笔数	>200	170-200	100-170	<100
进销规模比值	>0.5	0.4-0.5	0.1-0.4	<0.1
价税总额增长率	>1	0.5-1	0.1-0.5	<0.1
退款总额增长率	<0.1	0.1-0.3	0.3-1	>1
企业税率平均值	>0.13	0.1-0.13	0.09-0.1	<0.09

表 5-2 可以得到，信誉级别为 ABCD 的企业，在确定的九项指标下分别对应的区间，由此，可以根据计算所得的指标值对于级别的隶属度，建立决策矩阵，从而根据评价模型，得到相应的信誉级别。

5.3.2 基于 TOPSIS 的风险指数的建立

构建决策矩阵为：

$$D_{m\times n}=\begin{array}{c} \\ A_1 \\ A_2 \\ \cdots \\ A_m \end{array}\begin{array}{c} \begin{array}{cccc} X_1 & X_2 & \cdots & X_n \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \end{array}$$

Step1：对矩阵进行标准化处理，计算结果如下：

$$r_{ij}=\frac{x_{ij}}{\sqrt{\sum_{i=1}^{m}x_{ij}^2}}$$

Step2：构建决策矩阵：

$$V=\begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1j} & \cdots & v_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ v_{i1} & v_{i2} & \cdots & v_{ij} & \cdots & v_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ v_{m1} & v_{m2} & \cdots & v_{mj} & \cdots & v_{mn} \end{bmatrix}=\begin{bmatrix} w_1r_{11} & w_2r_{12} & \cdots & w_jr_{1j} & \cdots & w_nr_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_1r_{i1} & w_2r_{i2} & \cdots & w_jr_{ij} & \cdots & w_nr_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_1r_{m1} & w_2r_{m2} & \cdots & w_jr_{mj} & \cdots & w_nr_{mn} \end{bmatrix}$$

Step3：根据距离确定理想解：

$$A^+=\left\{\left(\max v_{ij}\mid j\in J\right),\left(\min v_{ij}\mid j\in J'\right)\mid i\in M\right\}=\left\{v_1^+,v_2^+,...,v_j^+,...,v_n^+\right\}$$

$$A^-=\left\{\left(\max v_{ij}\mid j\in J\right),\left(\min v_{ij}\mid j\in J'\right)\mid i\in M\right\}=\left\{v_1^-,v_2^-,...,v_j^-,...,v_n^-\right\}$$

Step4：计算指标到理想解的距离，本文选用欧氏距离，计算公式为：

$$S_{i^+}=\sqrt{\sum_{j=1}^{n}\left(v_{ij}-v_j^+\right)^2},i\in M$$

$$S_{i^-}=\sqrt{\sum_{j=1}^{n}\left(v_{ij}-v_j^-\right)^2},i\in M$$

Step5：计算相对贴切度：

$$C_{i^+}=\frac{S_{i^-}}{S_{i^-}+S_{i^+}},0<C_{i^+}<1,i\in M$$

由于本项指标采用的是正向评估，因此值越大代表风险指数越低。在本方案中，将确定的九项影响指标值带入计算，求得所有企业的信贷风险评估指数，举例部分企业如表 5-3 所示，所有结果见附件图：

表 5-3 部分企业信贷风险指数值

A 级企业	**E1**	**E2**	**E6**	**E7**
得分(0.9-1)	0.973	0.956	0.943	0.957
B 级企业	**E5**	**E10**	**E12**	**E23**
得分(0.3-0.4)	0.332	0.343	0.332	0.328

C 级企业	E3	E4	E11	E14
得分(0.1-0.2)	0.111	0.112	0.110	0.111
D 级企业	**E36**	**E52**	**E82**	**E99**
得分(<0.1)	0.006	0.0003	0.0001	0.0008

分析表 5-3 可知，使用 TOPSIS 建立的企业信贷风险评估指数具有很强的适用性和准确性，A 级企业信誉级别最高，指数位于 0.9 左右，B 级企业指数位于 0.3 左右，C 级企业指数位于 0.1 左右，D 级企业指数远远小于 0.1。同时，此时将企业风险级别数值化，为后文对于信贷策略选择提供数据支持。

5.4 基于目标优化的双阶段信贷策略选择模型

银行在年度信贷总额固定的情况下，制定对于信贷风险不同的企业的信贷策略，即可以转化为求解出银行对于信贷级别不同的企业每家的贷款金额和贷款利率。因此，本方案建立基于目标优化的双阶段信贷策略选择模型，第一阶段确定贷款金额，第二阶段确定贷款利率。根据题意可知，银行向实力强、供求关系稳定的企业提供贷款，并对信誉高、信贷风险小的企业给予利率优惠，即确定了两个决策变量。

5.4.1 阶段一：基于单目标优化的贷款金额选择策略

根据银行贷款定价的一般原则：利润最大化、扩大市场份额和保证贷款安全性，因此，银行对于企业的贷款金额，需要满足安全性高的企业相对多，安全性低的企业相对少，同时保证自身利益的最大化。综合以上考虑，本阶段，以贷款金额作为决策变量，以企业信贷风险评估指数作为决策变量的系数，以风险最小，即分数最大作为目标函数，可以得到目标函数的表达式为：

$$\max C = \sum_{i=1}^{n} c_i x_i$$

其中，C 代表所有企业的信贷风险评估指数，风险得分越高，风险越小。c_i 代表公司的风险评估分数。随后确定约束条件，公司的贷款金额为 10w-100w 之间，因此得到约束条件 1：

$$100000 \le x_i \le 1000000, i = 1, ..., n$$

由于金额的总数是固定的，得到约束条件 2：

$$\sum_{i=1}^{n} x_i = m$$

其中，m 为贷款额度总值。由此可以得到优化模型为：

$$\max C = \sum_{i=1}^{n} c_i x_i$$

$$s.t.\begin{cases}100000 \le x_i \le 1000000, i = 1,...,n \\ \sum_{i=1}^{n} x_i = m\end{cases}$$

假设此时贷款额度总值为 3000w 元，带入优化模型求解，由于题目说明银行对信誉评级为 D 的企业原则上不予放贷，所以计算是忽略 D 级企业，最终求解得到 ABC 级企业的贷款额度分别为 100w 元，40w 元和 10w 元，即 ABC 级企业的贷款额度以 10:4:1 的比例进行分配，得到如图 5-4 所示结果。

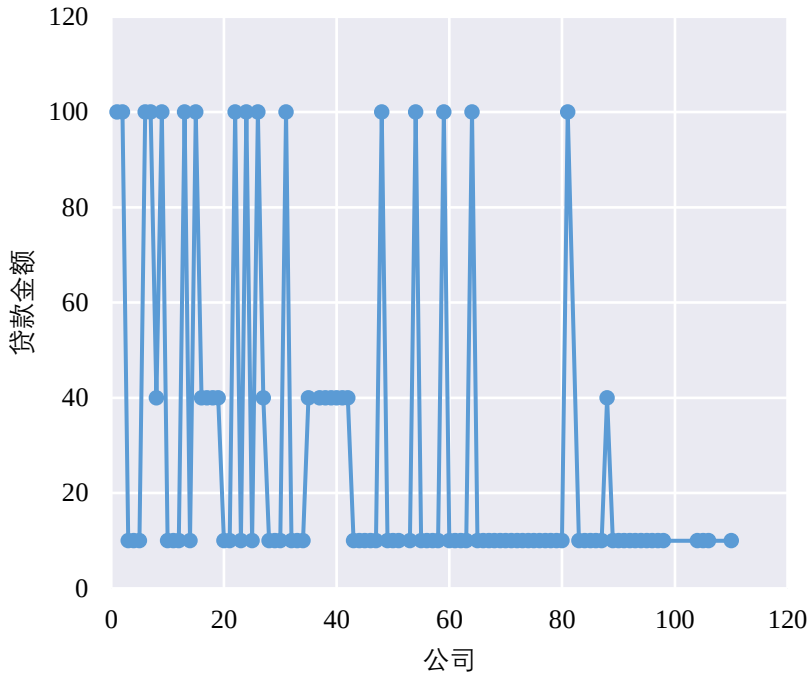

图 5-4 贷款金额分布图

5.4.2 阶段二：基于回归拟合的贷款利率选择策略

在确定了不同企业的贷款金额之后，基于贷款金额和不同信誉评级的企业的客户流失率[3]，需要确定此时最优的贷款年利率。因此，本阶段建立基于回归拟合的贷款利率选择模型，分为以下三个步骤：

Step1：贷款利率与客户流失率分析

根据附件三数据可知贷款利率和客户流失率在不同的等级下流失度是不同的，将相关数据进行图形绘制可得结果如图 5-5 所示。

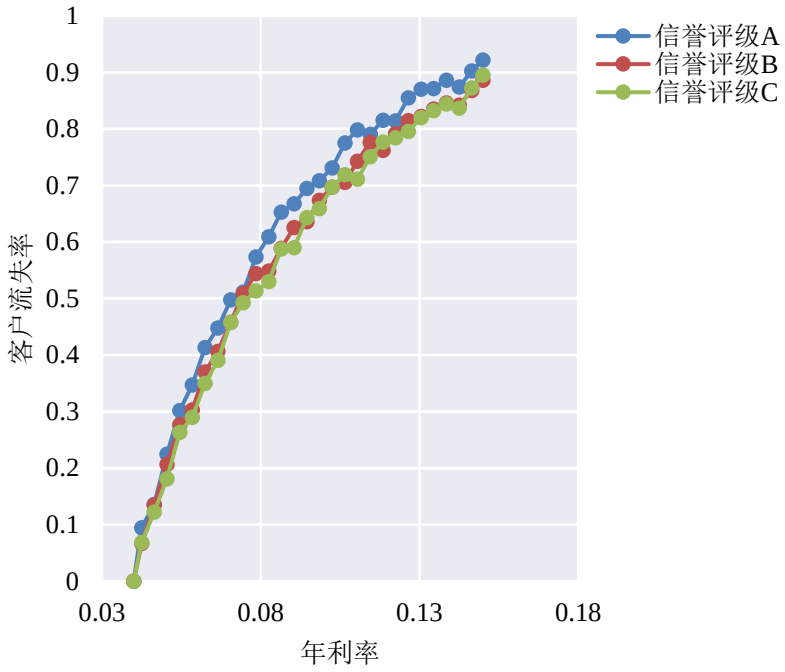

图 5-5 贷款利率与客户流失率关系图

由图 5-5 客户流失率与贷款利率的走势图，我们推测贷款利率与客户流失率呈指数关系或者线性关系，且对于信誉评级不同的企业，两者走势大体相同，所以可以使用同一模型进行拟合。

Step2：拟合贷款利率与客户流失率

分别采用指数关系和线性关系[4]进行拟合，根据经验及文献分析，设定指数拟合函数为：

$$y = a + b\ln x$$

由此，以信誉评级 A 的客户为例，得到基于指数关系拟合关系，如图 5-6 所示。

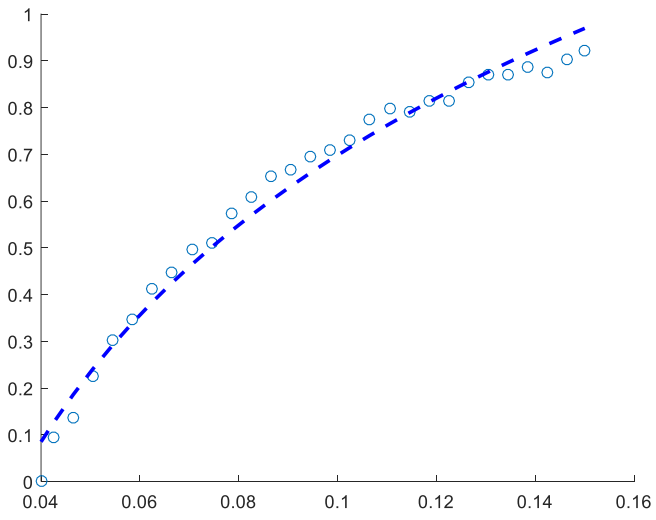

图 5-6 函数拟合图

分析图 5-6 可知，指数拟合比线性拟合更具有代表性，能够最大限度的拟合出贷款年利率和客户流失率的变化关系。接着，在拟合统计的基础上，本文得到指数函数和对数函数的 R 方分别为 0.896 和 0.982。综合以上两点分析，择优选择对数函数对函数进行拟合，并得到贷款利率和客户流失的函数关系为：

$$y_1 = 2.2386 + 0.669\ln x_1$$
$$y_2 = 2.1576 + 0.65057\ln x_2$$
$$y_3 = 2.168 + 0.65856\ln x_3$$

其中 y1、y2 和 y3 分别代表信誉评级 A、B 和 C 的客户流失率，x1、x2 和 x3 代表信誉评级 A、B 和 C 的贷款年利率。

Step3：基于银行期望收益最大化的最优年利率选择

当贷款金额确定时，银行对于放贷企业的期望收益可以表示为：

$$\text{期望收益}p_i = \text{贷款年利率}x_i \times \left(1 - \text{客户流失率}y_i\right)$$

即，对于信誉级别 A 的企业而言

$$p_1 = x_1\left(1 - 2.2386 - 0.699\ln x_1\right)$$

由于此时贷款金额时确定的，所以想要保证银行的总体利润最大化，即满足银行放出的每一笔贷款收益最大化，因此，求解式上式当 v 取最大值时，x 的取值即可满足要求。可视化式走势如图 5.7 所示。

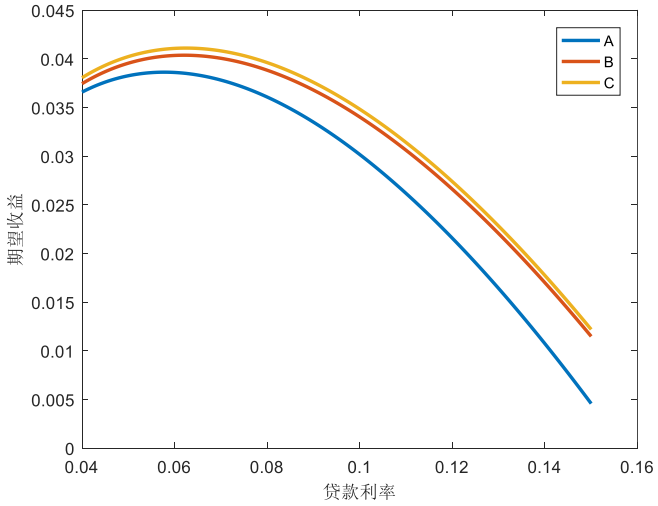

图 5-7 期望收益函数图

分析图 5-7 可知，通过计算韩式的拐点即可确定公司对于 A 级企业的期望最大收益。同样的方法运用于 BC 级企业，最终得到信誉评级 A、B 和 C 的企业最优年利率为 5.8%、6.2%和 6.3%。

6、问题二模型的建立与求解

问题二要求对附件2中302家企业的信贷风险进行量化分析，根据问题一的求解，本文首先对附件二数据进行数据预处理，计算确定的企业信誉等级的九项指标值；接着，建立基于熵权法的TOPSIS评价模型预测企业的信贷风险等级；最后，利用问题一建立的基于目标优化的双阶段信贷策略选择模型，对于银行贷款策略进行决策。

6.1 基于熵权法的 TOPSIS 评价模型企业信贷风险量化

由于问题一中建立的九项指标值对于企业信誉的影响程度不相同，所以首先需要对影响因素进行权重分析，再利用问题一中的量化结果对附件二企业进行信誉级别进行 TOPSIS 评价。TOPSIS 评价模型在 5.3.2 中已经描述，考虑到存在多项影响指标，所以本文利用熵权法从客观上求解指标权重。

6.1.1 熵权法确定指标权重

考虑到表 5-1 中各影响企业信誉指数的因素有九种，属于多因素权重确立，因此本文使用熵权法计算各因素权重值。利用以上九种影响因素作为评价对象，当评价对象在某项指标上的值相差较大时，熵值较小，说明该指标提供的信息量较大，该指标的权重也应较大。可用信息熵理论评价各指标的有序性及其效用，即由评价指标值构成的判断矩阵确定各评价指标权重。

(1) 数据标准化

将各个指标的数据进行标准化处理。假设有 n 项指标$\{C_1, C_2, \dots, C_n\}$，得到 m 组指标数据，由此得到数据集 A,即：

$$\mathrm{A} = \{a_{ij}\}, (i = 1, \dots, n; j = 1, \dots, m)$$

根据标准化公式：

$$\tilde{a}_{ij} = \frac{\tilde{a}_{ij} - \min(C_j)}{max(C_j) - \min(C_j)}$$

得到标准化后的数据集$\tilde{A} = \{\tilde{a}_{ij}\}, (i = 1, \dots, n; j = 1, \dots, m)$

(2)求各指标的信息熵

根据信息论中信息熵的定义，一组数据的信息熵为：

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

其中：

$$p_{ij} = \frac{\tilde{a}_{ij}}{\sum_{i=1}^{n} \tilde{a}_{ij}}$$

如果$p_{ij} = 0$，则定义$\lim_{p_{ij}=0} p_{ij} ln p_{ij} = 0$；

(3) 确定各指标权重

根据信息熵的计算公式，计算出各个指标的信息熵E_i,通过信息熵计算各指标权重即为：

$$\omega_i = \frac{1 - E_i}{n - \sum E_i} (\mathrm{i} = 1, \ldots, \mathrm{n})$$

6.1.2 模型的求解

(1) 数据预处理

根据问题一确定的九项企业信誉级别评价指标，对附件二的数据进行预处理，本文以“个体经营E124”为例，得到其12个月的数据预处理之后的结果如表6-1所示：

表 6-1 数据预处理结果表

	入账总额	大单交易额	大单交易额增长率	进总数	销总数	进销规模比值	价税总额增长率	退款总额增长率	税率均值
1	-29271150.7	3.19E+08	0.502925	50	155	0.322581	0.504659	0.001511	0.17
2	511995902.8	1.61E+08	3.987873	52	93	0.55914	3.93604	317.7843	
3	-224454645	6.4E+08	0.410964	55	146	0.376712	0.412912	0.811558	
4	127278317.3	2.63E+08	1.385213	54	127	0.425197	1.383687	0.687454	
5	329026520.2	3.65E+08	1.039751	57	139	0.410072	1.037711	1.49195	
6	15859821.3	3.79E+08	0.822045	47	152	0.309211	0.826231	0.471568	
7	208043442.7	3.12E+08	1.135536	69	168	0.410714	1.136254	0.607035	
8	321141624.5	3.54E+08	1.871061	80	165	0.484848	1.860864	6.354181	
9	70738364.9	6.62E+08	0.549062	104	164	0.634146	0.555151	1.626247	
10	354559240.2	3.64E+08	1.586619	100	174	0.574713	1.575812	0.563905	
11	166766324.2	5.77E+08	0.606002	65	178	0.365169	0.610854	0.061007	
12	-355992989	3.5E+08	0.555151	76	206	0.368932	0.546846	0	

(2) 基于熵权法的 TOPSIS 综合评价企业风险量化

由熵权法得到指标的权重向量为：

$$W = (0.2, 0.13, 0.06, 0.04, 0.02, 0.11, 0.17, 0.15, 0.12)$$

将指标选择权重带入TOPSIS评价的函数，根据调研确定正向理想解为[0.0211 ,0.0053,0.0105,0.0526 ,0.0105,0.1,10]，即可确定企业的信贷风险评估指数，得到各企业的信贷风险评估指数如下图6-1所示：

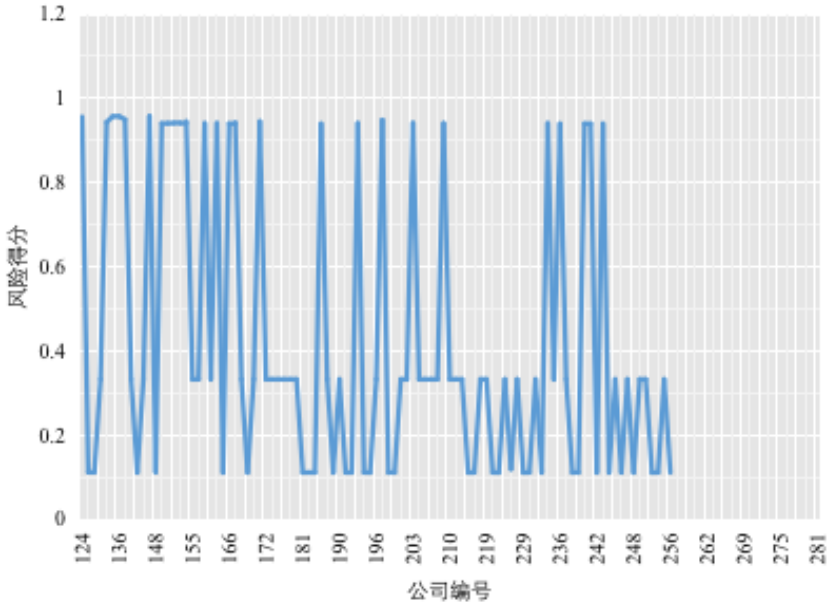

图 6-1 企业的信贷风险评估指数分布

根据表5-2确定模糊评价标准，根据企业的信贷风险评估指数，可以得到其信誉评级的量化结果为表6-2：

表 6-2 信誉评级量化结果表

信誉级别	相关企业编号
A级	124/136/142/145/146/149/151/152/154/162/165…416/418，共103个企业
B级	125/131/138/148/155/157/163/170/172/173/178…424/425，共127个企业
C级	126/129/132/133/134/135/137/139/140/141/144…337/373，共47个企业
D级	127/128/130/143/150/158/168/186/220/235/257…422/423，共20个企业

6.1.3 模型的检验

考虑到该问题是一个典型的四分类问题，同时属于有监督学习，因此本文采用LightGBM启发式机器学习算法验证算法的准确性。LightGBM是微软 2015 年提出的新的boosting框架模型，该算法在传统的 GBDT 基础上引入了两个新技术: 梯度单边采样技术和独立特征合并技术，具有更快的训练速度和更高的效率、更低的内存占用、相比于其他任何提升算法更高的准确率。最终，得到LightGBM和TOPSIS评价模型的分级对比图6-2。

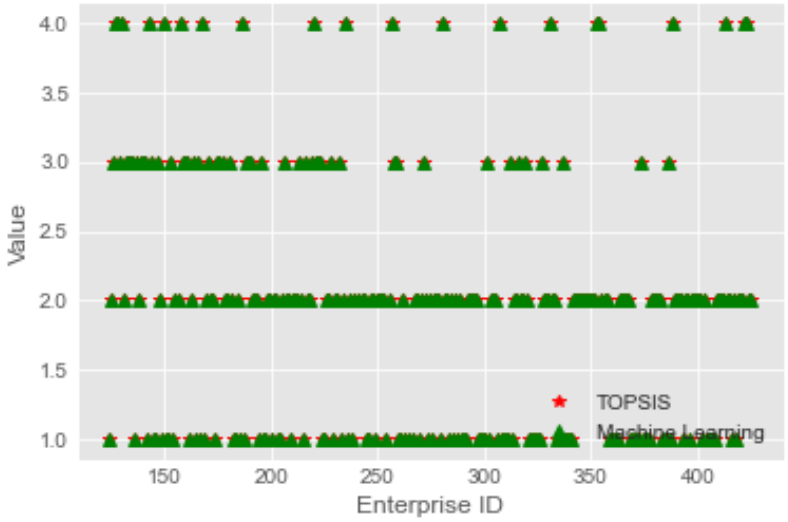

图 6-2 LightGBM 和 TOPSIS 评价模型的分级对比图

此时，LightGBM训练的结果与本文量化模型训练的结果有高达99.5%的匹配度，在图6-2中，红色标记代表本文量化结果，绿色标记代表机器学习分类结果，图中两者高度匹配，仅存在极少数异常情况，以此可以证明本文量化模型具有很强的分级准确性和适用性。

6.2 信贷策略决策

由于问题二此时，各企业的信誉评级结果已经在6.1中得到量化，所以该问题即为问题一中信贷策略决策的具体化，已知银行的年度信贷总额为1亿元，对于可以放贷的企业，贷款额度为10~100万元；年利率为4%~15%；贷款期限为1年。直接利用问题一种建立的基于目标优化的双阶段信贷策略选择模型求解即可。

将此时贷款额度总值为 1 亿元，带入优化模型求解，由于题目说明银行对信誉评级为 D 的企业原则上不予放贷，所以计算是忽略 D 级企业。建立决策变量为贷款金额的单目标优化模型：

$$\max C = \sum_{i=1}^{n} c_i x_i$$

$$s.t. \begin{cases} 100000 \le x_i \le 1000000, i = 1, ..., n \\ \sum_{i=1}^{n} x_i = 1亿 \end{cases}$$

最终，算法收敛的结果为：对所有A级企业和79.8%的B级企业放贷100元，对所有C级企业及21.2%的B级企业放贷10w元，分布结果如图6-3所示：

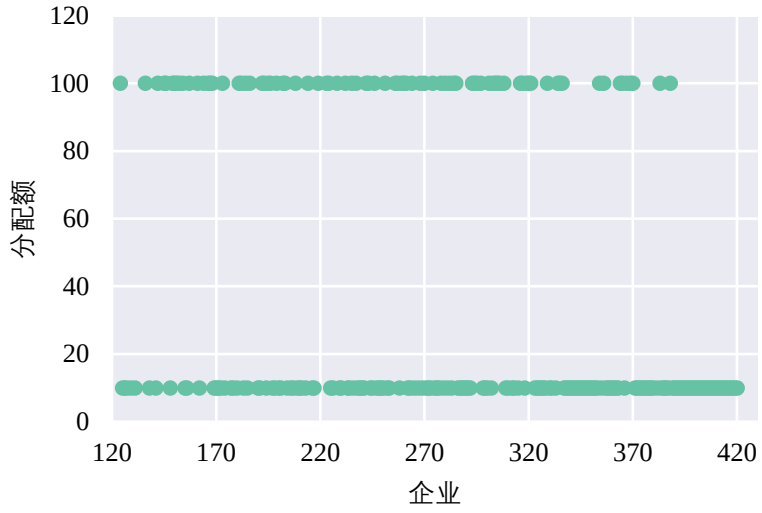

图 6-3 企业获得贷款金额分布图

此时，贷款利率计算为信誉评级A、B和C的企业最优年利率为5.8%、6.2%和6.3%。

6.3 基于多因素方差分析的模型验证

在前文的分析中，本文确定了影响因素之间对于企业信誉评级的影响是有差异的。因此，在这一部分我们选择采用多因素方差分析来控制水平验证因素差异之间的显著性。多因素方差分析[9]模型步骤如下：

对多个试验因素进行考察，A因素有i个不同水平的 $A_1, A_2, ..., A_n$ ，各因素之间无交互作用，对水平的每种组合 (A_n, B_m) 进行一次独立试验，共得出 $m \times n$ 个试验结果。

设 X 为服从正态分布 $X_{ij} \sim N(\mu_{ij}, \sigma^2)$ 的总体抽取的样本，假定

$$x_{ij} = \mu_{ij} + \varepsilon_{ij}$$

其中， μ_{ij} 表示 $A_i B_j$ 条件下的理论期望值， ε_{ij} 表示随机误差且相互独立。

$$\mu = \frac{1}{m \cdot n} \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_{ij}$$

$$\begin{cases} \mu_j = \frac{1}{n} \cdot \sum_{i=1}^{n} \mu_{ij} \\ \mu_i = \frac{1}{m} \cdot \sum_{j=1}^{m} \mu_{ij} \end{cases}$$

令 $\alpha_i = \mu_i - \mu, \beta_j = \mu_j - \mu$ ，称 α_i 为因素 A_i 的第i个水平的效应， β_j 为因素 B_j 的第 j 个水平效应， α_i ， β_j 分别表示因素A，B的各个水平的影响程度。令：

$$\sigma_{ij} = \mu_{ij} - \mu_i - \mu_j + \mu$$

式中， σ_{ij} 为 A_i 和 B_j 的交互效应，因为 A_i 和 B_j 之间无交互效应，所以 σ_{ij} =0，因此得到：

$$\mu_{ij}=\mu+\beta_j+\alpha_i$$

综上，可得出多因素无重复试验，试验方差的数学模型：

$$\begin{cases} x_{ij}=\alpha_i+\beta_j+\varepsilon_i \\ \sum_{i=1}^{n}\alpha_i=0,\sum_{j=1}^{m}\beta_j=0 \\ \varepsilon_{ij}\sim N\left(0,\sigma^2\right)\text{相互独立} \end{cases}$$

因而可以得到单因素方差分析数学模型：

$$Z_{ij}=\mu+\alpha_{1i}+\beta_{1j}+\alpha_{2i}+\beta_{2j}+\ldots+\alpha_{ni}+\beta_{nj}+\varepsilon_{ij}$$

其中，μ，σ^2，α，β 均为未知参数。

显著性检验为：

$$F_A=\frac{SS_A\big/(n-1)}{SS_B\big/(n-1)\cdot(m-1)}\sim F\left[n-1,(n-1)\cdot(m-1)\right]$$

其中，SS_A 为因素 A的效应平方和，SS_B 为因素 B 的效应平方和，对给定的显著性水平 α ，查F分布表得临界值 $F\left[n-1,(n-1)\cdot(m-1)\right]$，$F_n$ 取值通过查阅 F检验临界值表可知，当 $F_A>F_n$ 时，拒绝原假设，因素A影响显著；反之，接受原假设，因素 A影响不显著。

将处理后的指标数据进行显著性检验，选择置信度为95%，将SPSS结果统计输出如表 6.3 所示。

表 6-3 显著性检验表

影响指标	显著性水平
入账总额	0.034
大单交易额	0.028
大单交易额增长率	0
进项总笔数	0
销项总笔数	0
进销规模比值	0.045
价税总额增长率	0
退款总额增长率	0
企业税率平均值	0.032

由显著性检验表可知，所选的影响指标均通过了显著性差异，可以接受原假设，同时其显著性水平也与熵权法确定的指标权重相对应，验证了结果的可靠性。

7、问题三模型的建立与求解

企业的正常生产经营活动可能会受到一些突发事件的影响，而且不同行业、不同类别企业所受的影响也各不相同。突发事件一方面通过对企业的现金流、生

产经营活动等方面产生直接影响，一方面通过对整体行业、银行信贷政策等方面产生间接影响。突发状况发生后，由于企业正常生产经营活动与资金流情况发生波动，中小微企业的贷款偿还能力将受到损害，因此银行有必要对突发情况下信贷政策做出调整，以使得信贷风险[5]最小化。

7.1 基于动态蛛网理论的突发情况信贷调整模型

7.1.1 蛛网模型

信贷市场[6]是一个复杂的动态变化市场，放贷方与贷款方存在着复杂的博弈。过高的利率将会造成客户的过度流失，使得银行利益受损；而过低的利率将会吸引到较劣质客户，造成坏账率上升与银行利益损失。因此需要构建动态博弈模型准确量化放贷方与贷款方之间的博弈过程，求解出最优平衡点作为最优信贷方案。

蛛网模型[7]作为一种能够有效反映双边博弈的动态模型，能够有效模拟突发情况发生后放贷方与贷款方之间的动态博弈过程。

根据资金供给和资金需求弹性的关系，蛛网模型有三种分类：（1）收敛型：供给价格需求弹性小于需求价格弹性，市场偏离均衡价格之后，波动幅度将逐渐变小，最后回到初始均衡点；（2）发散型：供给价格弹性大于需求价格弹性，市场偏离均衡价格后将慢慢远离；（3）封闭型：供给价格弹性与需求价格弹性相等，市场围绕均衡点波动，且波动幅度保持一致。因此，根据需求量和供给量之间的动态关系，蛛网模型具体公式如下：

$$\begin{cases} Q_t^d = \alpha - \beta P_t \\ Q_t^s = -\delta + \gamma P_{t-1} \\ Q_t^d = Q_s^t \end{cases}$$

其中，Q_t^s 表示商品第 t 期的的供给量，Q_t^d 表示商品第 t 期的需求量，P_t 表示商品第 t 期的价格。α、β、δ、γ 为常数，均大于零。

7.1.2 信贷博弈动态蛛网模型

信贷商品[8]涉及到资金的供给方—放贷方与资金的需求方—贷款方之间的动态博弈过程，所以蛛网模型适用于信贷市场。根据题目条件可以合理假设短期内年度信贷总额不变。则突发情况对信贷市场的作用途径主要是通过改变加大需求曲线的价格弹性，在供给曲线价格弹性不变时，使得贷款供需逐渐回归均衡点。

因此引入突发情况因素[10]，对蛛网模型进行改进，得到信贷博弈动态蛛网模型如下所示：

$$\begin{cases} Q_t^d = \alpha - \beta(P_t + P_r) + cY \\ Q_t^s = -\delta + \gamma P_{t-1} \\ Q_t^d = Q_s^t \end{cases}$$

其中，Q_t^s 表示信贷资金第 t 期的的供给量，Q_t^d 表示信贷资金第 t 期的需求量，P_t 表示信贷资金第 t 期的价格，Y 表示人均可支配收入，P_r 表示突发情况导致的额外成本。α、β、δ、γ 为常数，均大于零。

假设短期内供给曲线保持不变，根据价格收敛条件，有：

（1）当 $(1-\delta\beta)^2 < 4\delta\gamma$ 时，信贷产品价格 P_t 收敛。

（2）当$(1-\delta\beta)^2 \geq 4\delta\gamma$时，在满足$\frac{2}{\beta-\gamma} \leq \delta \leq 0$的情况下，价格收敛。

可以发现，突发情况提高了贷款方的资金成本，即通过放大P_r取值的方法来降低贷款需求，稳定信贷产品市场。

7.2 模型的求解

（1）确定最优参数

本小节以 2019 年 12 月—2020 年 7 月间新冠疫情经济数据为例，探究突发情况对信贷市场的影响，以及求解最优信贷调整方案。

选取 2019 年 12 月-2020 年 7 月间新冠疫情经济数据作为分析，经济指标的选取分别为规模以上工业增加值环比增速(%)、固定资产投资环比增速(%)和社会消费品零售总额环比增速(%)，将其进行可视化图形绘制如图 7.1 所示。

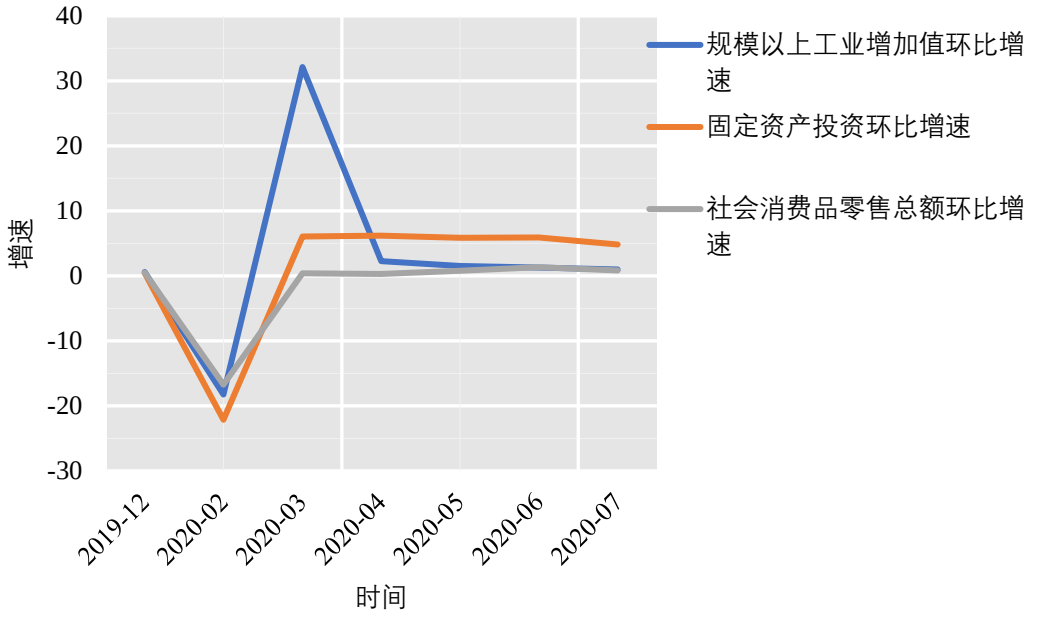

图 7-1 经济数据可视化

接着，利用最大似然估计法对需求和供给曲线进行回归，得到参数拟合值，结果见表 7-1：

表 7-1 参数拟合结果表

参数	参数拟合值
α	1516.34
β	0.375
c	0.156
δ	-1334.86
γ	0.703

由上表可以发现，新冠疫情情况下信贷市场供给曲线弹性与需求曲线弹性不相等，其中供给曲线弹性为$\gamma = 0.703$，需求曲线弹性为$\beta = 0.375$。供给曲线弹性

大于需求曲线弹性，属于蛛网模型中的收敛型。

将新冠疫情下信贷市场需求与供给曲线拟合参数代入价格收敛条件，解得银行在年度信贷总额为 1 亿元时的信贷策略为图 7.2 所示。

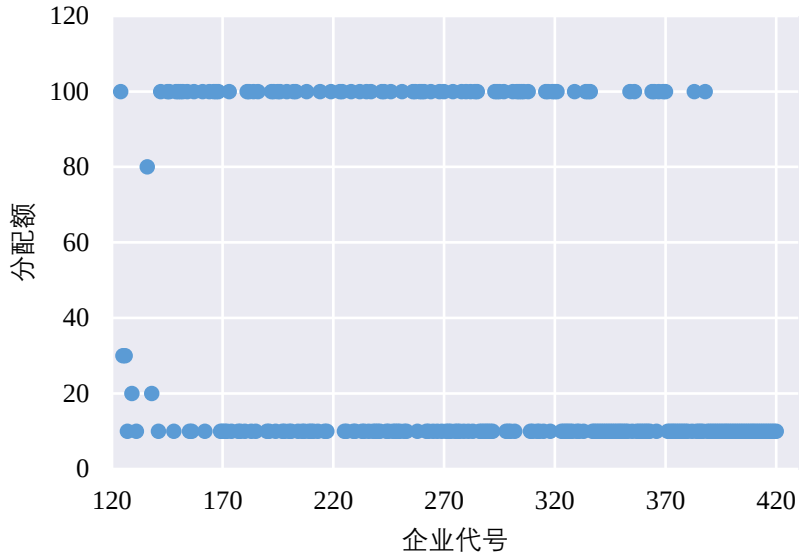

图 7-2 企业贷款分配额

8、模型的优缺点

8.1 模型优点分析

1、本文构建关于企业信誉指数的指标体系较为完善，且均通过了显著性检验，能够在很大程度上衡量企业信贷的风险性。

2、本方案采用两种不同的评价方式从定性和定量两个方面进行企业信贷风险的量化，可以为企业做信贷决策提供支持。

3、本文建立了基于目标优化的双阶段信贷策略选择模型，与机器学习结果匹配度达99.5%，说明模型具有很强的准确性和适用性。

4、本文建立了基于动态蛛网理论的突发状况调整模型，全面且精确的模拟出放贷方与贷款方之间的动态博弈过程，具有很强的实用价值。

8.2 模型缺点分析

1、本文基于附件所给出的相关数据进行分析，如果实际情况与相关数据有差异，那么模型所求结果也会有一定的误差。

2、实际影响企业信誉的因素较多，本文只选取了相关重要的因素。

8.3 模型改进方向

1、本文对于企业信贷指数影响因素的确定还偏少，并没有完全挖掘出表格中数据的潜在特征，因此，对于指标的选取还有改进空间

2、本文在信贷策略决策时，仅考虑了贷款金额和贷款利率的决策，对于还款期限没有限制，因此，可以增加约束条件，决策各级企业的还款期限。

9、参考文献

[1] 傅钰,池仁勇.征信要素、信用环境对中小微企业信用赋能的影响[J].现代管理科学,2020,(2):56-58.

[2] 牛燕影,王增富.基于 RR-EP 模型的风险量化技术[J].系统工程理论与实践,2013,33(05):1141-1148.

[3] 于小兵,曹杰,巩在武.客户流失问题研究综述[J].计算机集成制造系统,2012,18(10):2253-2263.

[4] 贺本岚.支持向量机模型在银行客户流失预测中的应用研究[J].金融论坛,2014,19(09):70-74.

[5] 兰军,严广乐.社会资本视角下中小企业信贷风险研究[J].中国流通经济,2019,33(05):111-119.

[6] 王蕾,郭芮佳,池国华.银行内部控制质量如何影响信贷风险?——基于行业风险识别视角的实证分析[J].中南财经政法大学学报,2019(04):3-12+158.

[7] Levieuge. Explaining and forecasting bank loans. Good times and crisis[J]. Applied Economics,2017,49(8).

[8] Gavalas,Syriopoulos. An integrated credit rating and loan quality model: application to bank shipping finance[J]. Maritime Policy & Management,2015,42(6).

[9] Comprehensive Credit Strategy, an option for Small and Medium-sized Enterprises in the Ecuador[J]. Revista Universidad y Sociedad,2018,10(2).

[10] Nagaraj V. Dharwadkar, Priyanka S. Patil. Customer retention and credit risk analysis using ANN, SVM and DNN[J]. Int. J. of Society Systems Science,2018,10(4).

10、附录

附录1：附件一中企业信贷风险指数分布

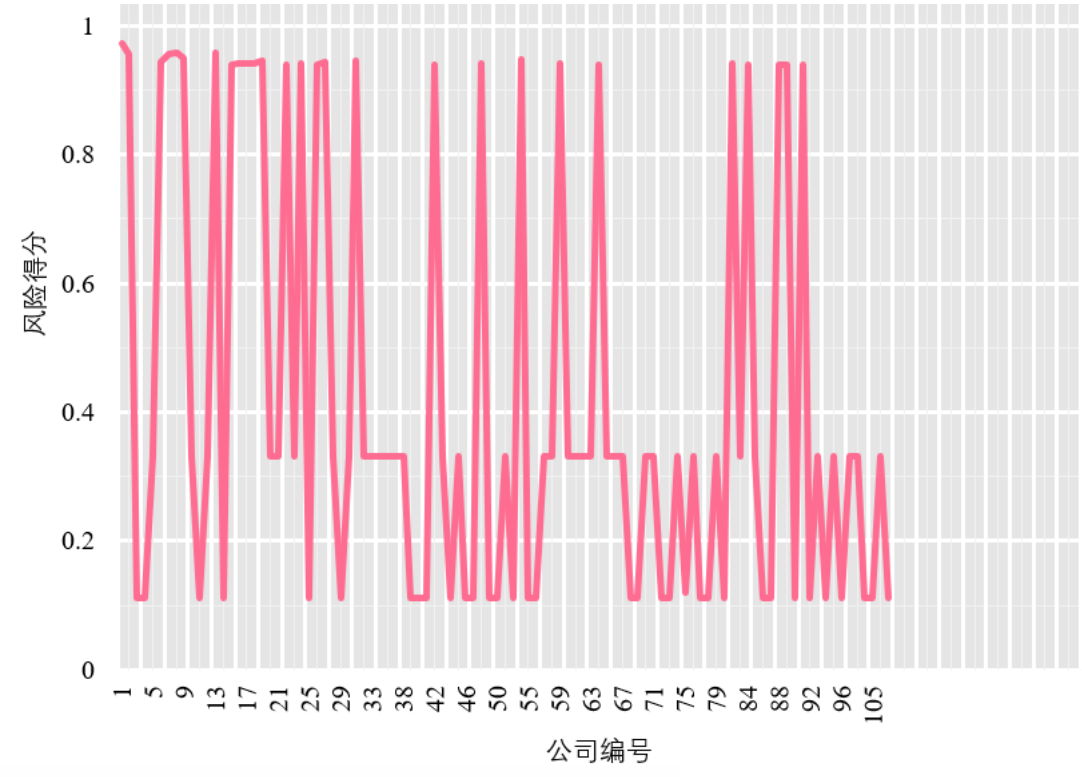

图 1 企业信贷风险指数分布

附录2：相关数据表格

C	D	E	F	G	H	I	J	K
企业代号	month	价税总额	大单交易额	大单交易额增长率	退款总额	退款总额增长率	退款大单交易额	数量
E1	1	324215052	319306064.2	0.502925177	-27524876.83	0.00151136	-27266561.38	155
E1	2	163617903	160587059	3.987873472	-41600	317.7842546	0	93
E1	3	644006627	640400872.6	0.410964189	-13219824.99	0.811558484	-13147427.15	146
E1	4	265918234	263181825.4	1.385212519	-1072661.13	0.68745423	-10647262.53	127
E1	5	367947537	364562759.2	1.039750684	-7375463.48	1.491949756	-7325463.48	139
E1	6	381823299	379054378.3	0.822044508	-11003820.94	0.471568198	-11003820.94	152
E1	7	315474352	311599570	1.13553607	-5189052.01	0.607035253	-5074720.01	168
E1	8	358459045	353832551	1.871060514	-3149937.5	6.354181345	-2732223.5	165
E1	9	667043652	662042114.9	0.54906182	-20015274.1	1.626247251	-19797293.1	164
E1	10	370309952	363502048.2	1.586618561	-32549784.49	0.563904535	-32488168.49	174
E1	11	583539004	576739096.6	0.60600227	-18354971.09	0.06100659	-18334477.09	178
E1	12	356457290	349505201.8	0.555150871	-1119774.2	0	-1004728.7	206
E10	1	194927406	194028117.3	0.101506646	0		0	17
E10	2	19695143.3	19695143.32	0.134460207	0		0	8
E10	3	2648213.05	2648213.05	0.735517484	-600000		-600000	4
E10	4	2067807	1947807	3.949035372	0		0	3
E10	5	7691958.74	7691958.74	1.954940384	0		0	7
E10	6	15037320.8	15037320.77	0.995679456	0		0	11
E10	7	15054688.5	14972351.36	2.346377378	0		0	9
E10	8	35339024.1	35130786.52	0.447268405	-472379.04		-472379.04	12
E10	9	15822471.9	15712890.85	0.864640259	0		0	10
E10	10	13645998	13585998.01	3.074996839	0		0	9
E10	11	41971900.9	41776900.94	0.40000327	0		0	13
E10	12	16796619	16710896.98	0	0		0	9
E100	1	72511	0	0	0		0	5
E100	2	13249.51	0	0	0		0	3
E100	3	48041	0	0	-4978.51		0	4
E100	4	66602.85	0	0	0		0	4

图2 销项发票处理数据

B	C	D	E	F	G	H	I	J
企业代号	month	价税合计	大单交易额	大单交易额增	退款总额	退款总额增长率	退款大单交易	数量
E1	2	294943901.4	294809376	2.291288588	-40000	1787.47095	0	52
E1	3	675613805.8	675493359	0.62079316	-71498838	0.002884606	-71498838	55
E1	4	419551982.3	419341657.2	0.937237176	-206246	0	-205920	54
E1	5	393196551.1	393022590.5	1.772724487	0	#DIV/0!	0	57
E1	6	696974057.2	696720770	0.568398038	0	#DIV/0!	0	47
E1	7	397683120.1	396014719	1.317615089	-30062829.05	2.328466156	-30000000	69
E1	8	523517794.3	521794969.2	1.298829634	-70000280	1.285723429	-70000000	80
E1	9	679600669.1	677722769	1.085129992	-90001000	0.888879012	-90000000	104
E1	10	737782016.8	735417303	0.983598812	-80000000	1.5	-80000000	100
E1	11	724869191.8	723355585.4	1.033032176	-120000000	0.685571775	-120000000	65
E1	12	750305328	747249594.5	0	-82268613	0.00040307	-82251333	76
E10	1	464300.29	0	#DIV/0!	-33160	0	0	107
E10	2	254247.9	0	#DIV/0!	0	#DIV/0!	0	92
E10	3	404603.46	0	#DIV/0!	-4536	12.51514771	0	128
E10	4	342996.64	0	#DIV/0!	-56768.71	0.006411983	0	143
E10	5	349603.22	0	#DIV/0!	-364	15.5267033	0	138
E10	6	667496.72	0	#DIV/0!	-5651.72	0.141387047	0	121
E10	7	529571.24	0	#DIV/0!	-799.08	13.55431246	0	156
E10	8	568370.52	0	#DIV/0!	-10830.98	0	0	163
E10	9	587754.74	0	#DIV/0!	0	#DIV/0!	0	186
E10	10	404868.66	0	#DIV/0!	-34844.72	0.852342909	0	202
E10	11	613477.93	0	#DIV/0!	-29699.65	0	0	184
E10	12	230670.32	0	#DIV/0!	0	#DIV/0!	0	112
E100	1	2733	0	#DIV/0!	0	#DIV/0!	0	6
E100	2	2306	0	#DIV/0!	0	#DIV/0!	0	5
E100	3	395	0	#DIV/0!	0	#DIV/0!	0	2
E100	4	1657	0	#DIV/0!	0	#DIV/0!	0	3
E100	5	2412.02	0	#DIV/0!	0	#DIV/0!	0	6
E100	6	3189	0	#DIV/0!	0	#DIV/0!	0	4
E100	7	763	0	#DIV/0!	0	#DIV/0!	0	1
E100	8	5972	0	#DIV/0!	0	#DIV/0!	0	2
E100	9	130	0	#DIV/0!	0	#DIV/0!	0	1
E100	10	937	0	#DIV/0!	0	#DIV/0!	0	3
E100	11	659	0	#DIV/0!	0	#DIV/0!	0	3
E100	12	2170	0	#DIV/0!	0	#DIV/0!	0	6

图3 进项发票处理数据

信誉评级	公司	得分	排序	信誉评级	公司	得分	排序
A	1	0.973678	1	B	5	0.332365	34
A	2	0.956209	5	B	10	0.332387	31
A	6	0.943466	10	B	12	0.332347	37
A	7	0.957148	4	B	20	0.332356	36
A	8	0.957676	2	B	21	0.332377	33
A	9	0.949636	6	B	23	0.332565	28
A	13	0.957411	3	B	28	0.332309	55
A	15	0.940425	21	B	30	0.332315	50
A	16	0.940981	15	B	32	0.332308	56
A	17	0.941858	12	B	33	0.332338	40
A	18	0.941045	14	B	34	0.33233	44
A	19	0.945506	8	B	35	0.332332	43
A	22	0.940485	20	B	37	0.332382	32
A	24	0.940868	17	B	38	0.332309	54
A	26	0.93994	22	B	43	0.332301	63
A	27	0.942821	11	B	45	0.332314	51
A	31	0.944939	9	B	51	0.332395	30
A	42	0.939552	26	B	57	0.332309	53
A	48	0.940896	16	B	58	0.332307	59
A	54	0.948791	7	B	60	0.332301	64
A	59	0.941243	13	B	61	0.332335	42
A	64	0.940608	19	B	62	0.332306	60
A	81	0.940666	18	B	63	0.332329	45
A	84	0.939652	25	B	65	0.332338	39

信誉评级	公司	得分	排序	信誉评级	公司	得分	排序
C	3	0.111347	70	D	36	0.006129	100
C	4	0.112273	67	D	52	0.0003	106
C	11	0.11087	75	D	82	0.000196	109
C	14	0.111885	69	D	99	8.18E-05	115
C	25	0.1109	72	D	100	0.00029	107
C	29	0.110854	85	D	101	9.7E-06	122
C	39	0.110854	88	D	102	0.000767	102
C	40	0.110941	71	D	103	0.000184	110
C	41	0.110864	78	D	107	4.37E-05	118
C	44	0.110865	77	D	108	0.000135	112
C	46	0.110888	73	D	109	0.000111	113
C	47	0.112162	68	D	111	0.000625	104
C	49	0.110876	74	D	112	5.75E-05	117
C	50	0.110853	92	D	113	3.99E-05	120
C	53	0.110858	81	D	114	0.000253	108
C	55	0.110855	84	D	115	5.57E-07	123
C	56	0.110869	76	D	116	0.000141	111
C	68	0.110852	96	D	117	4.32E-05	119
C	69	0.110854	86	D	118	0.000756	103
C	72	0.110853	93	D	119	0.000101	114
C	73	0.110861	79	D	120	3.35E-05	121
C	75	0.119249	66	D	121	0.000805	101
C	77	0.110852	95	D	122	0.000547	105
C	78	0.110859	80	D	123	7.38E-05	116

图4 附件1各级企业量化得分

由于表格量大，全部表格见支撑材料

附录3：关键步骤解题代码

```
#!/usr/bin/env python
# coding: utf-8
# In[1]:

import pandas as pd
import numpy as np
import datetime
data_notnull_num = 60          ##非 nan 最少条数要求


# In[2]:


#显示所有列
pd.set_option('display.max_columns', None)
#显示所有行
pd.set_option('display.max_rows', None)


# dt = pd.read_csv('input/rpt_email_core_usr_match_d_cjg_0801.csv')
# direction = pd.read_csv('direction/rpt.rpt_email_core_usr_match_d.csv')

dt = pd.read_csv('input/rpt_email_core_usr_grw_d_cjg_0801.csv')
direction = pd.read_csv('direction/rpt.rpt_email_core_usr_grw_d.csv')



# In[3]:


data_delta_index1 = list(chain_index_set['前几日'].unique())
month_delta_index1 = list(deviation_index_set['前几月'].unique())
extreme_delta_index1 = list(extreme_index_set['时间范围'].unique())


# In[4]:
```

```
for i in dt.columns:
    if ((direction[direction['字段名']==i]['字段说明'].isnull())==False).iloc[0]:
        dt.rename(columns={i:direction[direction['字段名']==i]['字段说明'].iloc[0]},inplace=True)
    elif ((direction[direction['字段名']==i]['字段说明'].isnull())==True).iloc[0]:
        dt.drop([i],axis=1,inplace=True)
dt.rename(columns={'date':'day'},inplace=True)
dt.day = pd.to_datetime(dt.day)


# In[5]:


conclusion_data_columns = ['day']
for data_delta_i in data_delta_index1:
    s = 'day_minus_'+str(data_delta_i)
    conclusion_data_columns.append(s)
conclusion_data_columns.append('备注')
conclusion_data = pd.DataFrame(columns=conclusion_data_columns)


# In[6]:


def deal_data(data_1,data_2,dt_delta,absolute_percent):

    data_2_columns = data_2.columns[0]
    data = pd.concat([data_1,data_2],axis=1)

    ## 环比指标/同比指标（相对百分比）
    res_1 = pd.DataFrame(index=[data_2_columns])

    T_raw = data.day.max()
    T = pd.to_datetime(T_raw)-datetime.timedelta(days=(dt_delta))
    T_data = (data[data['day']==T].iloc[0,1])
    res_1['day'] = T_data

    for data_delta_i in data_delta_index1:
        s= pd.to_datetime(T)-datetime.timedelta(days=(np.int(data_delta_i)))
        locals() ['T_minus'+str(data_delta_i)+'_data'] = (data[data['day']==s].iloc[0,1])
        res_1['day_minus_'+str(data_delta_i)] = (data[data['day']==s].iloc[0,1])
        T_minus_data = locals() ['T_minus'+str(data_delta_i)+'_data']
        for index_i in range(chain_index_set.shape[0]):
            chain_index_FieldName = chain_index_set.loc[index_i,'字段名称']
```

```
                chain_index_IndexName = chain_index_set.loc[index_i,'指标名称']
                chain_index_UpperThreshold = chain_index_set.loc[index_i,'阈值上限']
                chain_index_LowerThreshold = chain_index_set.loc[index_i,'阈值下限']
                chain_index_DayAgo = chain_index_set.loc[index_i,'前几日']
                if
(T_data>=T_minus_data*chain_index_LowerThreshold)&(T_data<T_minus_data*chain_index_
UpperThreshold)                 &(chain_index_IndexName !=-999)&(chain_index_DayAgo ==
data_delta_i):
                    res_1[chain_index_FieldName]                                    =
chain_index_IndexName+'{:.2%}'.format(T_data/T_minus_data-1)
                elif
(T_data>=T_minus_data*chain_index_LowerThreshold)&(T_data<T_minus_data*chain_index_
UpperThreshold)                 &(chain_index_IndexName ==-999)&(chain_index_DayAgo ==
data_delta_i):
                    res_1[chain_index_FieldName] = -999

    # # 环比指标/同比指标（绝对百分数）
    if absolute_percent==True:
        for data_delta_i in data_delta_index1:
            s= pd.to_datetime(T)-datetime.timedelta(days=(np.int(data_delta_i)))
            locals() ['T_minus'+str(data_delta_i)+'_data'] = (data[data['day']==s].iloc[0,1])
            res_1['day_minus_'+str(data_delta_i)] = (data[data['day']==s].iloc[0,1])
            T_minus_data = locals() ['T_minus'+str(data_delta_i)+'_data']
            for index_i in range(chainAbsolute_index_set.shape[0]):
                chain_index_FieldName = chainAbsolute_index_set.loc[index_i,'字段名称']
                chain_index_IndexName = chainAbsolute_index_set.loc[index_i,'指标名称']
                chain_index_UpperThreshold = chainAbsolute_index_set.loc[index_i,'阈值上
限']
                chain_index_LowerThreshold = chainAbsolute_index_set.loc[index_i,'阈值下
限']
                chain_index_DayAgo = chainAbsolute_index_set.loc[index_i,'前几日']
                if
(T_data>=T_minus_data+chain_index_LowerThreshold)&(T_data<T_minus_data+chain_index_
UpperThreshold)                 &(chain_index_IndexName !=-999)&(chain_index_DayAgo
== data_delta_i):
                    res_1[chain_index_FieldName]                                    =
chain_index_IndexName+'{:.2%}'.format(T_data-T_minus_data)
                elif
(T_data>=T_minus_data+chain_index_LowerThreshold)&(T_data<T_minus_data+chain_index_
UpperThreshold)                 &(chain_index_IndexName ==-999)&(chain_index_DayAgo
== data_delta_i):
                    res_1[chain_index_FieldName] = -999
```

```
    ## 偏离指标

    def Year_Month(year_input,month_input,month_delta):
        year_month_count = year_input * 12 + month_input
        year_month_count_end = year_month_count - 1
        year_month_count_start = year_month_count - month_delta

        year_end = year_month_count_end // 12
        month_end = year_month_count_end % 12
        year_start = year_month_count_start // 12
        month_start = year_month_count_start % 12

        return year_end, month_end, year_start, month_start

    res_2 = pd.DataFrame(index=[data_2_columns])

    data['month'] = pd.to_datetime(data['day']).apply(lambda x:x.month)
    data['year'] = pd.to_datetime(data['day']).apply(lambda x:x.year)
    for month_delta_i in month_delta_index1:
        Year_input = data['year'].max()
        Month_input = data['month'].max()
        year_end, month_end, year_start, month_start =
Year_Month(Year_input,Month_input,month_delta_i)
        locals() ['miu_'+str(month_delta_i)] =
data[((data['month']<=month_end)&(data['year']==year_end))&((data['month']>=month_start)&(d
ata['year']==year_start))][data_2_columns].mean()
        locals() ['std_'+str(month_delta_i)] =
data[((data['month']<=month_end)&(data['year']==year_end))&((data['month']>=month_start)&(d
ata['year']==year_start))][data_2_columns].std()
        miu = locals() ['miu_'+str(month_delta_i)]
        std = locals() ['std_'+str(month_delta_i)]
        for index_i in range(deviation_index_set.shape[0]):
            deviation_index_FieldName = deviation_index_set.loc[index_i,'字段名称']
            deviation_index_IndexName = deviation_index_set.loc[index_i,'指标名称']
            deviation_index_UpperThreshold = deviation_index_set.loc[index_i,'阈值上限']
            deviation_index_LowerThreshold = deviation_index_set.loc[index_i,'阈值下限']
            deviation_index_MonthAgo = deviation_index_set.loc[index_i,'前几月']
            if
(T_data>=miu+deviation_index_LowerThreshold*std)&(T_data<miu+deviation_index_UpperThr
eshold*std) &(deviation_index_MonthAgo==month_delta_i):
                res_2[deviation_index_FieldName] = deviation_index_IndexName


    ## 极值指标
```

```
    res_3 = pd.DataFrame(index=[data_2_columns])
    for extreme_delta_i in extreme_delta_index1:
        s = pd.to_datetime(T)-datetime.timedelta(days=(np.int(extreme_delta_i)))
        s_max = (data[data['day']>=s].iloc[:,1].max())
        s_min = (data[data['day']>=s].iloc[:,1].min())
#           print(T_data,s_max,s_min,extreme_delta_i)
        for index_i in range(extreme_index_set.shape[0]):
            extreme_index_FieldName = extreme_index_set.loc[index_i,'字段名称']
            extreme_index_IndexName = extreme_index_set.loc[index_i,'指标名称']
            extreme_index_TimeRange = extreme_index_set.loc[index_i,'时间范围']
            extreme_index_Flag = extreme_index_set.loc[index_i,'flag(大 1 小-1)']
            if
(T_data>=s_max)&(extreme_delta_i==extreme_index_TimeRange)&(extreme_index_Flag==1):
                res_3[extreme_index_FieldName] = extreme_index_IndexName
            elif
(T_data<s_min)&(extreme_delta_i==extreme_index_TimeRange)&(extreme_index_Flag==-1):
                res_3[extreme_index_FieldName] = extreme_index_IndexName
            if
(T_data<s_max)&(T_data>=s_min)&(extreme_delta_i==extreme_index_TimeRange)&(extreme_
index_Flag==0):
                res_3[extreme_index_FieldName] = extreme_index_IndexName


    # # 连续指标
    res_4 = pd.DataFrame(index=[data_2_columns])

    cnt_increase = 0
    cnt_decrease = 0
    # T = data.day.max()
    for i in range(1,len(data)-10):
        delta = data[data['day']==pd.to_datetime(T)-datetime.timedelta(days=(i))].iloc[0,1] -
data[data['day']==pd.to_datetime(T)-datetime.timedelta(days=(i+1))].iloc[0,1]
        if delta >=0:
            cnt_increase += 1
        elif delta < 0:
            break
    for j in range(1,len(data)-10):
        delta = data[data['day']==pd.to_datetime(T)-datetime.timedelta(days=(j))].iloc[0,1] -
data[data['day']==pd.to_datetime(T)-datetime.timedelta(days=(j+1))].iloc[0,1]
        if delta < 0:
            cnt_decrease += 1
        elif delta >= 0:
            break
```

```
    for index_i in range(continuous_index_set.shape[0]):
        continuous_index_FieldName = continuous_index_set.loc[index_i,'字段名称']
        continuous_index_IndexName = continuous_index_set.loc[index_i,'指标名称']
        continuous_index_Threshold = continuous_index_set.loc[index_i,'阈值（希望连续大于多少天则预警）']
        continuous_index_Flag = continuous_index_set.loc[index_i,'flag(涨 1 降-1)']
        if (cnt_increase>=continuous_index_Threshold)&(continuous_index_Flag==1):
            res_4[continuous_index_FieldName] = continuous_index_IndexName+str(cnt_increase)+'天'
        elif (cnt_increase<continuous_index_Threshold)&(continuous_index_Flag==1):
            res_4[continuous_index_FieldName] = -999
        elif (cnt_decrease>=continuous_index_Threshold)&(continuous_index_Flag==-1):
            res_4[continuous_index_FieldName] = continuous_index_IndexName+str(cnt_decrease)+'天'
        elif (cnt_decrease<continuous_index_Threshold)&(continuous_index_Flag==-1):
            res_4[continuous_index_FieldName] = -999

    ## 汇总指标
    res = pd.concat([res_1,res_2],axis=1)
    res = pd.concat([res,res_3],axis=1)
    res = pd.concat([res,res_4],axis=1)

    data_delta_columns_minus = []
    for data_delta_i in data_delta_index1:
        data_delta_columns_minus.append('day_minus_'+str(data_delta_i))
    data_delta_columns_minus.append('day')

    res_conclusion_index_before=res.columns
    res_conclusion_index_before = res_conclusion_index_before.drop(data_delta_columns_minus)
    res_conclusion_index_after = ''
    for i in res_conclusion_index_before:
        if res[i][0]!=-999:
            res_conclusion_index_after = res_conclusion_index_after+str(res[i][0]+', ')

    res_conclusion = res[data_delta_columns_minus].copy()
    res_conclusion['备注'] = res_conclusion_index_after

    return res_conclusion


# In[7]:
```

```
data_delta_columns_minus = ['day']
for data_delta_i in data_delta_index1:
    data_delta_columns_minus.append('day_minus_'+str(data_delta_i))
data_delta_columns_minus.append('备注')

data_1=dt[['day']]
for i in dt.columns:
    if i !='day':
        if ((direction[direction['字段说明']==i]['指标类型'])==3).iloc[0]:
            if '次日' in i:
                dt_delta_value = 1
            elif '7 日' in i:
                dt_delta_value = 7
            elif '30 日' in i:
                dt_delta_value = 30
            else:
                dt_delta_value = 0
            if dt[i].notnull().sum()>=data_notnull_num:
                dt[i].fillna('-99900%',inplace=True)
                dt[i] = dt[i].apply(lambda x:np.float(x.strip("%"))/100)
                absolute_percent = True
                data_2=dt[[i]]
                s=deal_data(data_1,data_2,dt_delta_value,absolute_percent)
                conclusion_data=pd.concat([conclusion_data,s],axis=0)
            elif dt[i].notnull().sum()<data_notnull_num:
                s = pd.DataFrame(columns=data_delta_columns_minus,index=[i])
                for i in conclusion_data.columns:
                    s[i]='有效数据量太少，不足'+str(data_notnull_num)+'条'
                conclusion_data=pd.concat([conclusion_data,s],axis=0)
        elif (((direction[direction['字段说明']==i]['指标类型'])==1).iloc[0])|(((direction[direction['字段说明']==i]['指标类型'])==2).iloc[0]):
            if '次日' in i:
                dt_delta_value = 1
            elif '7 日' in i:
                dt_delta_value = 7
            elif '30 日' in i:
                dt_delta_value = 30
            else:
                dt_delta_value = 0
            if dt[i].notnull().sum()>=data_notnull_num:
                dt[i].fillna(-999,inplace=True)
                absolute_percent = False
                data_2=dt[[i]]
                s=deal_data(data_1,data_2,dt_delta_value,absolute_percent)
```

```
            conclusion_data=pd.concat([conclusion_data,s],axis=0)
        elif dt[i].notnull().sum()<data_notnull_num:
            s = pd.DataFrame(columns=data_delta_columns_minus,index=[i])
            for i in conclusion_data.columns:
                s[i]='有效数据量太少，不足'+str(data_notnull_num)+'条'
            conclusion_data=pd.concat([conclusion_data,s],axis=0)



# In[8]:


conclusion_data.to_csv('output/conclusion_data.csv')


# In[9]:


conclusion_data


# In[26]:


def gen_user_nunique_features(df, value, prefix):
    group_df = df.groupby(['user']).agg(
        s=locals()['user_'+str(1)+'_'+str(1)+'_nuniq'],
        s=pd.NamedAgg(column=1, aggfunc='nunique'),
    ).reset_index()
    return group_df


# In[23]:


locals()['user_'+str(1)+'_'+str(1)+'_nuniq']=pd.NamedAgg(column=1, aggfunc='nunique'),


# In[ ]:
```