基于企业规模、现金流、销售质量及供求稳定性的银行信贷策略研究

摘要

随着京东、爱奇艺等寡头企业的不断壮大，越来越多的中小微企业倾向于模仿它们的发展路径：前期不断融资，以扩大市场占有率；进而逐步提高商品售价，实现公司盈利。因此，中小微企业的信贷需求显著增加，这要求银行需要制定准确、合理的信贷决策，以使自己的期望收益最大化。对此，我们为银行建立了信贷决策模型，综合考虑企业实力、信誉等指标，利用因子分析、二元 Logistic 回归、支持向量机及深度学习的方法，结合经济学的 RAROC 模型，通过数学规划手段求解出了银行在固定年度信贷总额下的最优信贷策略。

针对问题一　本文首先基于附件 1 中给出的数据，建立了从现金流、企业规模、销售质量和供求关系稳定性四方面综合评价企业实力和信誉的评价体系。接着，分析各指标特点，运用因子分析对指标进行降维与去量纲处理。此后，我们将企业的信贷风险量化为守约率，并建立了二元 Logistic 回归模型、SVM 支持向量机模型和全连接 BP 神经网络模型用于预测守约率，选择了三者中最优的模型——神经网络，进行后续预测。最后，结合客户流失率和企业的信贷风险，在年度信贷总额固定的情况下，建立了 RAROC 模型，通过确立目标函数和约束条件，给出了能够使银行期望收益最大的决策方案，称由此得到的放贷企业顺序为最优解序列，同时对信贷总额不同的情况进行了分类讨论。

针对问题二　不同于问题一，问题二要求我们基于无信贷记录企业的数据进行信贷决策。因此，本文首先引入模糊数学中隶属度的概念，建立了基于 BP 神经网络的四分类模型，得出了各企业对不同信誉等级的隶属度，并预测出各企业在不同利率等级下的流失率。接着，用问题一建立的二分类模型预测企业的信贷风险，并基于第一问模型，结合客户流失率，将问题转化为一个数学规划问题，进而求解出在信贷总额为 1 亿的前提下，银行贷款发放的最优解序列。

针对问题三　我们首先将企业按照对风险的敏感性划分为三类：风险敏感型企业、风险稳定型企业和风险受益型企业，并计算出不同类型的企业遭遇风险时的主要业务的风险损失率。接着，考察风险对企业净现金流增长率，现金流波动性，销售总额，净税，利润率等主要指标的影响，结合风险损失率更新原有放贷策略，获得更新的最优解序列。进一步，我们通过更改风险强度参数，得出了不同风险强度对企业的影响，预测了面临不同强度的突发事件时，银行应该做出的策略改变。最后，我们考虑了突发事件发生时，国家在税收方面，不同程度的减免政策，进而调整银行放贷的策略。

本文从不同方面，综合考虑了影响企业实力及信誉的各种指标，建立了评估企业信贷风险的量化模型。利用 EXCEL 的数据透视表功能及 Python 中 xlrd 和 xlwt 公开库，进行了数据清洗及预处理，得到了各个企业守约率的具体数值。随后，基于 PyTorch，Scikit-Learn 等机器学习框架，求解了二分类和四分类预测模型，并基于数学规划求解出了银行的最优解序列。接着，本文综合考虑各种突发状况及政府应对负向突发状况的补贴措施，得出了银行在发生不同强度、不同性质的突发状况下，对应的最优解序列。最后，总结了模型的优缺点及可能的优化方向。

关键词　**因子分析**　**二元 Logistic 回归**　**支持向量机**　**BP 神经网络**　**数学规划模型**

1 问题重述

1.1 问题背景

近年来，中小微企业蓬勃发展，成为社会经济繁荣稳定的一注新鲜血液。与此同时，中小微企业也成了银行贷款的一个重要客户群体。银行给予中小微企业信用贷款在一定程度上促进了企业的发展。然而由于中小微企业规模相对较小，也缺少抵押资产，所以准确评估企业的信贷风险，并根据评估结果制定合理的信贷策略，是银行必须面对的问题。

为了建立科学的信贷风险评估体系，银行通常重点考察企业的交易票据信息和上下游企业的影响力，并依据信贷政策、向实力强、供求关系稳定的企业提供贷款，且对信誉高、信贷风险小的企业给予贷款利率优惠。首先，银行将根据中小微企业的实力、信誉评估其信贷风险，然后依据信贷风险等因素来确定信贷策略，如：是否放贷及贷款额度、利率和期限等。

某银行需要对不同的企业进行是否放贷、贷款额度及利率的决策。其中放贷年限为 1 年，放贷金额为 10~100 万元，而利率为 4%~15% 间的离散值。附件 1, 2 分别给出了不同数量的企业信息及其发票信息，主要的区别在于是否对企业记录了企业的违约状况及是否对企业进行了信用评级。附件 3 为贷款利率与客户流失率关系的 2019 年统计数据。本文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，主要解决下列问题：

(1) 基于附件 1 中提供的企业信誉及发票流水，量化企业的信贷风险，求解该银行在年度投资额固定时的信贷策略。

(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) 本文综合考虑附件 2 中各企业的信贷风险和可能的突发因素（包括正向因素和负向因素）对各企业的影响，并给出了不同突发情况下，该银行在年度信贷总额为 1 亿元时的信贷调整策略。

1.2 目标任务

基于上述背景，我们需要建立数学模型解决以下三个问题：

问题一　通过附件 1 中的 123 家企业的信用评级、违约记录和进销项发票数据，提出量化企业信贷风险的指标，并将这些指标用于预测企业的信贷风险。基于量化后的信贷风险，建立决策模型给银行提供期望收益最大的策略。

问题二　基于问题一建立的信贷风险量化评估模型，预测各企业的信贷风险。同时，建立预测企业信誉评级和不同利率下流失率的数学模型，进而确定在年度信贷总额为 1 亿元的条件下，制定银行对各企业的信贷策略。

问题三　在问题二的基础上，再将可能的突发因素对企业的影响考虑在内，并从企业的名称对企业分类，将不同行业加入信贷风险评估指标，对银行在年度信贷总额为 1 亿元时做出信贷策略的调整。

2 问题分析

通过对题目提供附件的综合分析可知，本题是一个针对银行的信贷决策问题。我们需要提取发票数据的关键信息，综合考虑企业实力、信誉以及不同利率下的流失率，做出能使**银行期望收益最大**的决策。此外，本题对于预测的精度要求较高，因此本文综合考虑了多种预测模型，选取最优模型。我们期望这个模型训练精度较高，同时有较低的可能性出现过拟合。

2.1 问题一的分析

在本问题中，要重点考虑如何**通过发票的指标量化企业的信贷风险**，即企业获得贷款后违约的概率。之后，综合考虑企业的信贷风险及不同利率下的企业流失率，在不同总金额的情况下做出期望收益最大的信贷决策。

2.2 问题二的分析

与问题一不同的是，问题二的数据没有提供企业的信誉等级，因此无法准确得到企业在不同利率下的流失率。为此，我们筛选了问题一中与企业信誉相关度较高的指标，建立了一个**四分类模型**，将附件一的数据作为训练集，用于预测附件二各企业关于各的隶属度，进而加权得到企业在不同利率下的流失率。之后，利用预测出的流失率进行信贷决策。

2.3 问题三的分析

本题要求增加突发因素等因素的影响，针对于不同行业的企业进行信贷策略调整。为此，我们对各企业的风险敏感程度进行了划分，综合评判了突发事件对企业各指标的影响，模拟了不同风险条件下企业的反应，最后综合考虑国家政策等因素对企业的正面影响。

3 模型假设

1. 公司票据如实给出，票据信息与实际情况相符，包含企业在相应时间内的所有交易记录，且不存在发票造假；
2. 银行对中小企业信贷决策是理性的，即在制定信贷策略的过程中以期望收益最大化为目标；
3. 问题一和问题二的求解过程中不考虑可能的突发因素对各企业的影响；
4. 处在相同行业的不同企业，受某突发状况的影响相同；
5. 国家对所有企业一视同仁，企业获得的税率减免百分比相等；
6. 企业在理性状态下做出判断，流失率完全符合附件 3 中的变化规律；
7. 不考虑除题目所给企业数据之外的其他因素对信贷风险评估的影响；
8. 允许银行对不同企业有不同的放贷利率，且由银行自行决定；
9. 不考虑银行放贷过程中产生的手续费和其他费用；
10. 当企业违约时，银行将无法使用任何方法获得补偿；
11. 遭遇风险时，仅影响企业的相应关键指标，而对其他指标如上下游企业供应稳定性无影响；
12. 银行可以自由选择贷出的金额比例，允许剩余一定的资金以防止可能的风险。

4 符号说明

表1列出了本文需要的符号。

表 1: 符号说明

符号	符号描述
N	待分析的企业总数
T	划分的时期总数
$IN_{i,t}$	第 i 家企业在第 t 个时间段的进项订单量
$OUT_{i,t}$	第 i 家企业在第 t 个时间段的销项订单量
$Money_{i,t,k}^{(in)}$	第 i 家企业在第 t 个时间段内的第 k 个进项订单金额（税后）
$Money_{i,t,k}^{(out)}$	第 i 家企业在第 t 个时间段内的第 k 个销项订单金额（税后）
$Money_{i,t}$	第 i 家企业在第 t 个时间段内的净现金流入（税后）
$Tax_{i,t,k}^{(in)}$	第 i 家企业在第 t 个时间段内的第 k 个进项订单的税金
$Tax_{i,t,k}^{(out)}$	第 i 家企业在第 t 个时间段内的第 k 个销项订单的税金
$Tax_{i,t}$	第 i 家企业在第 t 个时间段内的增值税总额
$X_i^{(j)}$	第 i 家企业的第 j 项指标（因子分析前）
x_{ij}	第 i 家企业的第 j 个信贷风险评估因子的得分（因子分析后）
σ_i	第 i 家企业的净现金流标准差
$\overline{Money_i}$	第 i 家企业在时间 T 内的平均现金流
Y_i	第 i 家企业是否违约的二分变量（0/1）
$level$	信誉评级，取值为 A,B,C,D
$level_i$	信誉评级位于 $level$ 类别的第 i 个企业的新编号
$RAROC_{level_i}$	银行从守约企业 $level_i$ 获得的利息收入
W_{level_i}	银行从守约企业 $level_i$ 收回的本金
$Capital_{total}$	银行年度信贷总额
In	银行贷款年利率
$Lose_{In,level_i}$	In 利率下，$level$ 等级的 i 企业的客户流失率
r_{level}	银行放贷给 $level$ 等级的企业的贷款额度占年度信贷总额的比率
α_i	风险敏感型/稳定型/收益型企业的风险损失率,$i = 1, 2, 3$

5 模型的建立

5.1 问题一模型的建立

题目中提供的数据包括四方面，分别是：

1. 企业进项发票信息；
2. 企业销项发票信息；
3. 企业的名称、信誉评级和违约情况；
4. 不同信誉评级企业的客户流失率随贷款年利率的变化的情况。

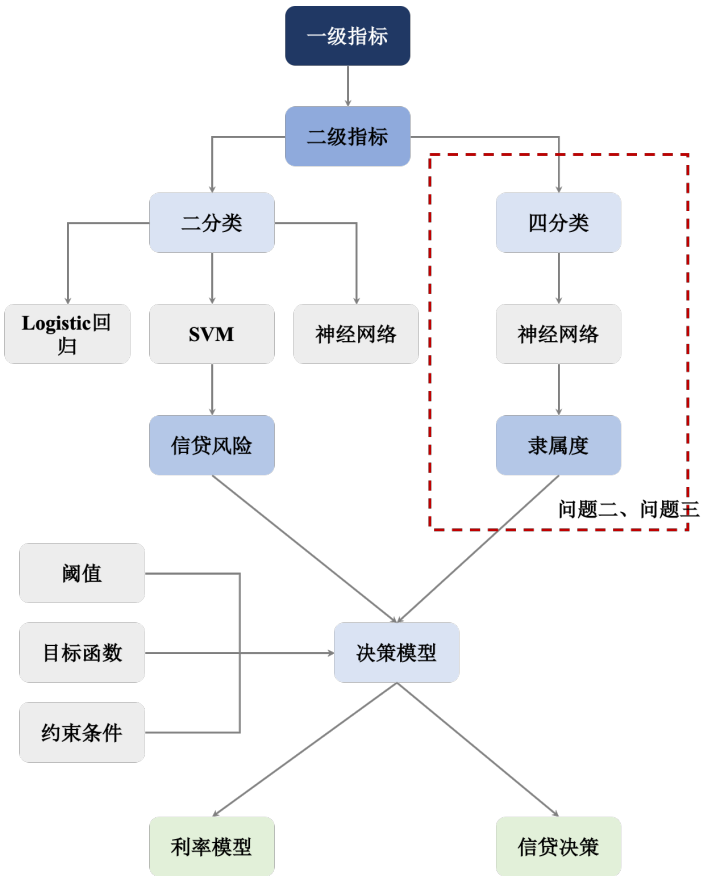


图 1: 本文建模流程图

5.1.1 指标选取原则

银行在考虑是否对某一企业放贷时，往往最看重企业的实力、信誉和信贷风险。我们需要基于题目提供的数据，选取合适的指标来评价一个企业，量化该企业的信贷风险。基于此，本文指标的选取遵循下列原则[4]。

（一）科学性原则

企业实力及信誉的评价指标背后应具有坚实的数理基础和确切的经济学含义，能够全面、准确地揭示企业的财务状况。评价体系指标中涉及到的各指标名称、含义、计量单位和计算方法，须符合客观实际的经济规律。

（二）可行性原则

评价体系指标的选取要具备可行性，包括可计量性和可操作性。其中，可计量性指各指标的定量计算必须通过科学且合理的手段。可操作性指确立的指标能够完全基于现有数据计算得来，具有可复现性。

（三）全面性原则

全面性原则要求指标的选取须统筹兼顾，在能够充分体现企业实力及信誉整体特性的前提下，不失企业间的特殊性。既应该选取多指标以满足全面性要求，又应避免指标冗余。

5.1.2 指标类型的确立

一个企业的现金流往往能很好的反应企业的综合实力，也能在一定程度上预测短期内的财务状况[13]。银行可以根据过去企业的现金流情况，初步判断企业发生资金链断裂的可能性，进而进行信

贷决策。

同时，企业的规模也是银行进行放贷决策时常考虑的重要因素[5]。一般来说，规模大的企业往往更加稳定，不容易破产，能够及时偿贷[1]。

除此之外，企业的销售质量也是应该纳入考虑。一个企业的销售质量高，往往会拥有稳定的客户，资金链发生意外的可能性也较低。

类似地，企业的上下游企业供求关系是否稳定也应该纳入考虑。从进货的角度来说，如果一家企业经常去某几家公司采购原材料，那么企业获得的原材料水平在时间维度上往往较为稳定，利于企业生产出质量稳定的产品；从销售的角度来说，如果企业拥有稳定的回头客，说明企业的产品质量得到了市场的认可，往往会给企业的发展带来正面影响。

综上，我们确定了四类衡量企业的指标，分别是**现金流指标**，**企业规模指标**，**企业销售质量指标**和**企业上下游供求关系稳定性指标**。图2简要地说明了本文确定的各个指标下面将阐述各个指标类别内具体指标的确立与计算方式。其中，为节省空间，我们将“企业在相同上游企业进货两年及以上的进货金额占比”简写成“上游企业两年回头客进货金额占比”，另外三项同理。

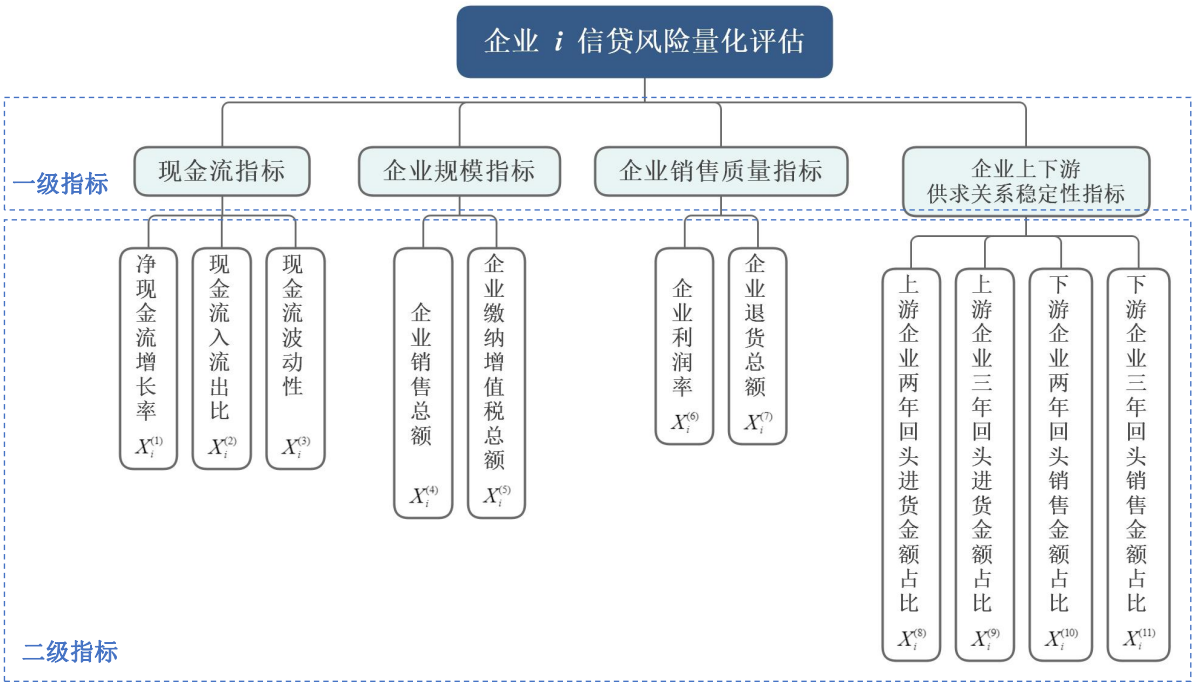


图 2: 指标说明

5.1.3 现金流指标

针对现金流的评价指标，一般可以归类为五类：获现能力，盈利能力，偿债能力，财务弹性和成长能力[8]。其中，获现能力强往往伴随着较高的盈利能力财务弹性和成长能力，而偿债能力也与财务弹性和成长能力相关。因此，判断企业的获现能力及偿债能力，能对企业的现金流进行初步的判断。

银行向企业贷款，本质上是企业的债权人，但是在某种程度上也可以理解为企业的投资人。投资人希望企业创造的价值能够稳定提高，同时获取可观利润。为此，我们首先考虑企业的发展潜力，设

[1]当然，大规模企业一旦破产，造成的损失也是巨大的。

立**净现金流增长率**指标，其计算公式为：

$$\text{净现金流增长率} = \frac{\text{本期经营活动现金流} - \text{上期经营活动现金流}}{\text{上期经营活动现金流}} \tag{1}$$

结合题目提供的发票数据，我们首先计算净现金流：

$$Money_{i,t} = \sum_{k=1}^{OUT_{i,t}} Money_{i,t,k}^{(out)} - \sum_{k=1}^{IN_{i,t}} Money_{i,t,k}^{(in)} \tag{2}$$

其中 $i = 1, 2, \cdots, N$ 为企业下标，$t = 1, 2, \cdots, T$ 为时间下标。这里 OUT, IN 表示企业货物的卖出（销项）与买入（进项）。$Money_{i,t}$ 表示第 i 家企业在时期 t 内的净现金流总额；$OUT_{i,t}$ 表示第 i 家企业在时期 t 内的销项发票数量，$Money_{i,t,k}^{(out)}$ 表示第 i 家企业在时期 t 内第 k 张销项发票的金额；$IN_{i,t}$ 表示第 i 家企业在时期 t 内的进项发票数量，$Money_{i,t,k}^{(in)}$ 表示第 i 家企业在时期 t 内第 k 张进项发票的金额。

结合式(1),(2)，我们可以计算得出企业 i 的净现金流增长率 $X_i^{(1)}$：

$$X_i^{(1)} = \frac{1}{T-1} \sum_{t=2}^{T} \frac{Money_{i,t} - Money_{i,t-1}}{Money_{i,t-1}} \tag{3}$$

除了企业的发展能力外，企业的**现金流入流出比**也是一个重要的指标。它具体的计算公式为：

$$\text{现金流入流出比} = \frac{\text{现金流入量}}{\text{现金流出量}} \tag{4}$$

该指标反映了企业能否通过自身的运营来维持企业的日常运营指标。该指标的临界值为 1，表示净现金流为 0。通常情况下，该指标越大越好，表示企业有足够的经营现金流（即企业可支配的现金流较充裕）；如果某企业的现金流入流出比长期小于 1，表明该净现金流长期 < 0（即长期亏损），因此需要贷款融资来维持企业的正常经营，长此以往容易造成资金链断裂而破产，从而导致信贷违约的发生。结合题目提供的发票数据以及式(2)和(4), 我们可以计算企业 i 的现金流入流出比 $X_i^{(2)}$：

$$X_i^{(2)} = \frac{1}{T} \sum_{t=1}^{T} \frac{\sum_{k=1}^{OUT_{i,t}} Money_{i,t,k}^{(out)}}{\sum_{k=1}^{IN_{i,t}} Money_{i,t,k}^{(in)}} \tag{5}$$

除此之外，企业的**现金流波动性**也在一定程度上反应了企业的供求关系是否稳定。理论上来看，波动越大，企业的稳定性越差，面临净现金流为负的风险也就越大[13]。本文用现金流的标准差来衡量其波动性，计算公式为：

$$\text{现金流标准差} = \sqrt{\frac{1}{T-1} \sum_{t=1}^{T} (\text{第 } t \text{ 期实际现金流} - \text{平均现金流})^2} \tag{6}$$

其中，平均现金流是指对时间做平均。联立(2)和(6), 我们可以计算出企业 i 的净现金流标准差 σ_i：

$$\overline{Money_i} = \frac{1}{T} \sum_{t=1}^{T} Money_{i,t} \tag{7}$$

$$\sigma_i = \sqrt{\frac{1}{T-1} \sum_{t=1}^{T} \left(Money_{i,t} - \overline{Money_i}\right)^2} \tag{8}$$

其中 $\overline{Money_i}$ 表示第 i 家企业在时间 T 内的平均净现金流。但是，由于不同企业的规模大小相差悬殊，导致标准差的量纲难以比较。且现金流波动性指标是一个负向指标（波动越小，稳定性越好），我们需

要将其转化为正向指标（数值越大越好）。本文参考[9]，采用取倒数的方法将其正向化。进一步，将标准差的倒数 $\frac{1}{\sigma}$ 乘以平均净现金流 $\overline{Money_i}$ 来消除量纲。由此，我们得到经过正向化、无量纲化处理的企业的现金流波动性指标：

$$X_i^{(3)} = \frac{\overline{Money_i}}{\sigma_i} \tag{9}$$

至此，我们已经确立了从不同层面评价企业现金流的三个指标：**净现金流增长率**主要体现企业的获现能力、盈利能力和成长能力，**现金流入流出比**主要体现企业的获现能力和偿债能力，**现金流波动性**主要体现企业的财务弹性和偿债能力。

5.1.4 企业规模指标

大部分企业通过销售总额这一指标来评判企业规模[12]。企业 i 的销售总额 $X_i^{(4)}$ 计算方法为：

$$X_i^{(4)} = \sum_{t=1}^{T} \sum_{k=1}^{OUT_{i,t}} Money_{i,t,k}^{(out)} \tag{10}$$

此外，企业缴纳的增值税也能够反映企业规模[17]。结合增值税的定义，在本题中，第 i 家企业在时期 t 下的增值税总额 $Tax_{i,t}$ 为：

$$Tax_{i,t} = \sum_{k=1}^{OUT_{i,t}} Tax_{i,t,k}^{(out)} - \sum_{k=1}^{IN_{i,t}} Tax_{i,t,k}^{(in)} \tag{11}$$

结合式(11)，企业的平均增值税 $X_i^{(5)}$ 计算方式如下：

$$X_i^{(5)} = \frac{1}{T} \sum_{t=1}^{T} Tax_{i,t} \tag{12}$$

至此，我们确立了两个指标来衡量一个企业的规模，分别是**企业销售总额**和**企业缴纳增值税总额**。这两个指标都能反映行业的规模，但是不同行业的增值税率有显著区别。一般情况下，烟草、化妆品等行业增值税较高[2]。综合判断销售总额和增值税总额，可以间接判断一个企业所处的行业。因此，这两个指标有必要同时纳入考量。

5.1.5 企业销售质量指标

企业的销售质量，一方面由企业的利润率体现，另一方面由企业商品的退货情况反映[6]。退货项在本题中为销项中金额为负数的发票。第 i 家企业的利润率 $X_i^{(6)}$ 的计算方式为：

$$X_i^{(6)} = \sum_{t=1}^{T} \frac{Money_{i,t}}{\sum_{k=1}^{IN_{i,t}} Money_{i,t,k}^{(in)}} \tag{13}$$

第 i 家企业的退货总额的计算方式为：

$$X_i^{(7)} = \sum_{t=1}^{T} \sum_{k=1}^{OUT_{i,t}} I_{\{Money_{i,t,k}^{(out)} < 0\}} \cdot Money_{i,t,k}^{(out)} \tag{14}$$

[2] https://baike.baidu.com/item/高税率产品/12803440?fr=aladdin

其中 I_P 表示命题 P 是否发生，定义如下：

$$I_P = \begin{cases} 1, & P \text{ 为真} \\ 0, & \text{其他} \end{cases} \tag{15}$$

至此，我们定义了企业的**利润率**和**退货总额**来反映企业的销售质量。其中，利润率更侧重于评判企业产品的单位利润，而退货总额则侧重于评判企业产品的质量稳定性。

5.1.6 企业上下游供求关系稳定性指标

企业的上下游供求关系稳定性主要由两方面构成：上游的稳定性，下游的稳定性。企业上游的稳定性越高，即企业经常去相同的原材料公司购买生产所需的待加工产品，那么一般情况下企业的产品质量也会比较稳定；企业下游的稳定性越高，即企业的顾客群体稳定，那么某种程度上能够说明企业的产品得到了顾客的认可，利于企业的长期发展[11]。

同时，题目提供数据的时间范围基本在 2017 年第三季度到 2019 年第四季度[3]，因此本文对于上游的供求稳定性确立了上游供应稳定性指标 $X_i^{(8)}$ 和 $X_i^{(9)}$，分别表示第 i 家企业在相同上游企业进货两年及以上的进货金额占比和三年及以上的进货金额占比，计算方式如下：

$$X_i^{(8)} = \frac{\sum_l \sum_{t=1}^{T} \sum_{k=1}^{IN_{i,t}} I_{\{\text{企业 } i \text{ 从公司 } l \text{ 进货两年及以上}\}} \cdot Money_{i,t,k}^{(in)}}{\sum_{t=1}^{T} \sum_{k=1}^{IN_{i,t}} Money_{i,t,k}^{(in)}} \tag{16}$$

$$X_i^{(9)} = \frac{\sum_l \sum_{t=1}^{T} \sum_{k=1}^{IN_{i,t}} I_{\{\text{企业 } i \text{ 从公司 } l \text{ 进货三年及以上}\}} \cdot Money_{i,t,k}^{(in)}}{\sum_{t=1}^{T} \sum_{k=1}^{IN_{i,t}} Money_{i,t,k}^{(in)}} \tag{17}$$

其中 I_P 的定义如式(15)所示。

类似地，对于下游的供求稳定性确立了下游需求稳定性指标 $X_i^{(10)}$ 和 $X_i^{(11)}$，分别表示第 i 家企业在相同下游企业销售两年及以上的销售金额占比和三年及以上的销售金额占比，计算方式如下：

$$X_i^{(10)} = \frac{\sum_l \sum_{t=1}^{T} \sum_{k=1}^{OUT_{i,t}} I_{\{\text{企业 } i \text{ 从公司 } l \text{ 销售两年及以上}\}} \cdot Money_{i,t,k}^{(out)}}{\sum_{t=1}^{T} \sum_{k=1}^{OUT_{i,t}} Money_{i,t,k}^{(out)}} \tag{18}$$

$$X_i^{(11)} = \frac{\sum_{t=1}^{T} \sum_{k=1}^{OUT_{i,t}} I_{\{\text{企业 } i \text{ 从公司 } l \text{ 销售三年及以上}\}} \cdot Money_{i,t,k}^{(out)}}{\sum_{t=1}^{T} \sum_{k=1}^{OUT_{i,t}} Money_{i,t,k}^{(out)}} \tag{19}$$

5.1.7 因子分析模型

在上文提出 11 个信贷风险量化评估二级指标（变量）之后，我们对数据进行 KMO 检验和巴特利特球形度检验，分别分析取样适切性量数和显著性来确定是否适合做因子分析。

若数据通过检验，我们计算变量间的相关系数矩阵来探究这些变量之间的内部依赖关系。用碎石检验（Scree Test）确定公共因子的个数，并用相应个数综合因子来表示其基本的数据结构。由于归结出的因子个数少于原始变量的个数，但是它们又包含原始变量的信息，所以因子分析模型能达到降维的目的，并且新的因子能反映原来众多变量的主要信息，和并有更容易解释的实际含义。

我们用**主成分分析法**计算因子载荷矩阵，得到初始公因子。由此确定的因子模型后，其中的公共因子不一定能反映问题的实质特征。为了能更好地解释每一个公共因子的实际意义，且减少解释的主观性，我们通过凯撒最大化最大方差法进行因子旋转（正交变换），使因子载荷矩阵的结构简化，使新公共因子的载荷系数的绝对值尽可能接近 0 或 1。

[3]2020 年仅有一月的数据，不利于分析供求稳定性，不予考虑。

因子旋转后，我们可以得到旋转成分矩阵和总方差解释表，从而得到每个公共因子所解释的方差及累计和，变量与公共因子的相关系数，进而计算出所有的公共因子得分，作为企业新的信贷风险量化评估指标的得分值。

5.1.8 二元 Logistic 回归模型

设上文中因子分析模型共得到 t 个公共因子, x_{ij} 表示第 i 个企业的第 j 个信贷风险评估因子的得分。由于第 i 个企业是否违约的变量 Y_i 是二分变量（0/1），本文采用以下二元 Logistic 回归模型。设 $\boldsymbol{\beta} = (\beta_0, \beta_1, \cdots, \beta_t)$ 是待确定的系数。我们在给定 $\boldsymbol{x}$ 的情况下，考虑 y 的两点分布概率:

$$\begin{cases} P(y=1|\boldsymbol{x}) = F(\boldsymbol{x}, \boldsymbol{\beta}) \\ P(y=0|\boldsymbol{x}) = 1 - F(\boldsymbol{x}, \boldsymbol{\beta}) \end{cases} \tag{20}$$

其中连接函数 $F(\boldsymbol{x}, \boldsymbol{\beta})$ 取 $Sigmoid$ 函数:

$$F(\boldsymbol{x}, \boldsymbol{\beta}) = S(\boldsymbol{x^T}, \boldsymbol{\beta}) = \frac{exp(\boldsymbol{x^T} \cdot \boldsymbol{\beta})}{1 + exp(\boldsymbol{x^T} \cdot \boldsymbol{\beta})} \tag{21}$$

由于

$$E(y|\boldsymbol{x}) = 1 \times P(y=1|\boldsymbol{x}) + 0 \times P(y=0|\boldsymbol{x}) = P(y=1|\boldsymbol{x}) \tag{22}$$

故可将 $\hat{y}$ 理解为事件 $y=1$(即企业不违约) 发生的概率。

$$\boldsymbol{x^T} \cdot \boldsymbol{\beta} = \beta_0 + \beta_1 \cdot x_{1i} + \beta_2 \cdot x_{2i} + \cdots + \beta_t \cdot x_{ti} \tag{23}$$

我们将数据随机分成两组：训练集和测试集。对训练集的数据作二元 Logistic 回归，利用极大似然估计法 (MLE) 得到 $\beta_0, \beta_1, \cdots, \beta_t$ 的估计值 $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_t$。再由

$$\begin{aligned} \hat{Y_i} = P(Y_i = 1|\boldsymbol{x}) = S(\boldsymbol{x_i^T} \cdot \hat{\boldsymbol{\beta}}) &= \frac{\exp(\boldsymbol{x_i^T} \cdot \hat{\boldsymbol{\beta}})}{1 + \exp(\boldsymbol{x_i^T} \cdot \hat{\boldsymbol{\beta}})} \\ &= \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \hat{\beta}_2 x_{2i} + \cdots + \hat{\beta}_k x_{ki}}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \hat{\beta}_2 x_{2i} + \cdots + \hat{\beta}_k x_{ki}}} \end{aligned} \tag{24}$$

计算出测试数据集中企业 i 的是否违约变量的估计 $\hat{Y_i}$ 值。$\hat{Y_i} \geq 0.5$, 则认为预测的 $Y_i = 1$, 表示第 i 家企业不违约; $\hat{Y_i} < 0.5$, 则认为预测的 $Y_i = 1$，表示第 i 家企业违约。

5.1.9 基于 SVM 的二分类预测模型

考虑到本问题样本量较小且已经完成一定特征工程，我们采用支持向量机（SVM）的方法来进行二分类。由于样本数量较少，而特征较多，我们采用**高斯核**进行实验。

1. 区分训练集和测试集
2. 通过对 γ 值的调参[4]
3. 检查是否过拟合

[4]调参过程详见问题一求解部分。

5.1.10 基于 BP 神经网络的二分类预测模型

确立了指标后，我们以附件 1 中**企业是否违约**为训练目标 Y_i，输入为 $\boldsymbol{x_i} = (x_{i,1},\ x_{i,2}, \cdots, x_{i,6})$，其中 $x_{i,j}$ 表示：经过因子分析后，第 i 家企业的第 j 项因子[5]。Y_i 的定义如下：

$$Y_i = \begin{cases} 1, & \text{企业 } i \text{ 未违约} \\ 0, & \text{企业 } i \text{ 违约} \end{cases} \tag{25}$$

本文利用了一个含一层隐藏层的全连接神经网络 (Fully-Connected Neural Network) 求解这个二分类问题[6]，其结构如图3所示. 输出层中 Y_0 表示企业违约的概率，Y_1 的意思恰好相反，表示企业遵守

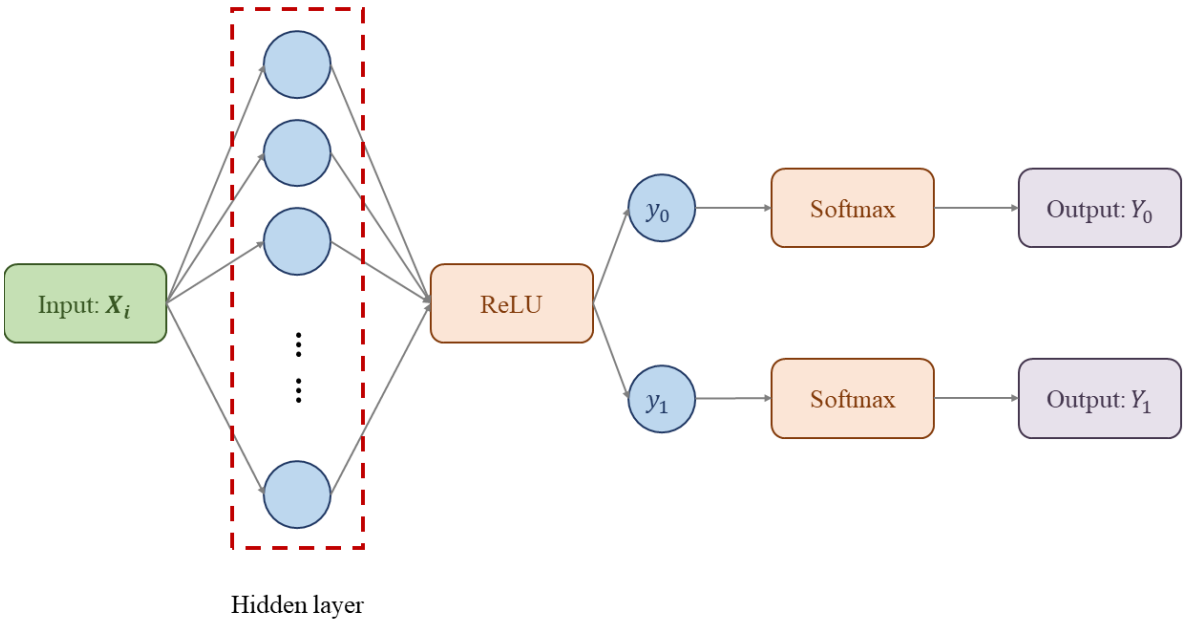


图 3: 全连接神经网络结构示意图

约定按期还款的概率（守约率）。

5.1.11 信贷决策模型

在求得每个企业的违约可能性之后，我们将建立贷款额度分配模型来计算给各企业分配的贷款额度以及收取的贷款利率。我们引入风险调整资本回报率 RAROC 模型[1]。在获得最高资本回报率的原则下，加入风险的影响。具体方法如下:

1. 设置阈值

银行将根据两个原因拒绝贷款给超高风险企业：

- 当企业的信贷风险评估分数低于某一阈值，即企业的信贷风险相当高，违约的概率相当大。银行将不会在任何利率下贷款给上述企业。
- 当企业的信誉评级为 D 类时，银行将不予以放贷。

[5]使用因子分析的原因详见问题一的求解部分。

[6]选择该网络结构的具体原因详见问题一的求解部分。

2. 确定目标函数

银行将通过对确定放贷的企业，贷款金额和贷款利率做出决策，以实现回报率的最大化。设通过第一步阈值检验的 A,B,C 类企业分别为 A_1,$A_2,\cdots,A_{num_A}$; B_1,$B_2,\cdots,B_{num_B}$; $C_1,C_2,\cdots,C_{num_C}$. 简单起见，我们在下文中用 $level_i(i=1,2,\cdots,num_{level};level=A,B,C)$ 表示位于 $level$ 类别的第 i 个企业。由**风险调整资本回报率 RAROC** 模型，我们确定如下目标函数:

$$RAROC_{total}=\frac{(\text{预期收入}R\text{—预期损失}P)}{\text{经济资本}Capital_{total}} \tag{26}$$

- 预期收入R = A,B,C 类企业的利息 + A,B,C 类企业的本金 − 银行年度信贷总额
- 预期损失P = 当企业违约时，银行损失的本金数额
- 经济资本$Capital_{total}$ = 银行年度信贷总额

剔除 D 等级后，得到表达式如下:

$$R=R_A+W_B+R_B+W_B+R_C+W_C-Capital_{total} \tag{27}$$

- $R_{level_i}(i=1,2,\cdots,num_{level};level=A,B,C)$ 表示银行从守约企业 $level_i$ 获得的利息收入。
- $W_{level}(i=1,2,\cdots,num_{level};level=A,B,C)$ 表示银行从守约企业 $level_i$ 收回的本金。

从银行的角度看，每个企业能够收回的有效本金的期望为有效投入资金与企业守约率的乘积。即

- E(收回的有效本金) = 客户保留率 × 获投资金 × (1 − P(企业违约率))
- E(收回的有效利息) = E(收回的有效本金) × 利率

其中 E 表示数学期望。

$$R_{level_i}=Capital_{total}\times r_{level_i}\times(1-Lose_{In,level_i})\times Y_{level_i}\times(1+In) \tag{28}$$

其中，

1. r_{level_i} 表示银行授予位于 $level$ 等级的 i 企业的放贷占银行放贷总额的比例。
2. $Lose_{In,level_i}$ 表示在 In 利率下，位于 $level$ 等级的 i 企业的客户流失率。
3. Y_{level_i} 表示位于 $level$ 等级的 i 企业的守约率。
4. In 表示银行决策的 $level$ 等级 i 企业本年度的放贷利率。
5. $level=A,B,C$ 表示企业的信誉评级有 A，B，C 三类，$level_i(i=1,2,\cdots,num_{level};level=A,B,C)$ 表示位于 $level$ 类别的第 i 个企业。

$$R_{level}=\sum_{i=1}^{num_{level}}R_i,\quad i=1,2,\cdots,num_{level};level=A,B,C \tag{29}$$

预期损失 P 为企业在选择向银行借贷时发生违约，无法偿还任何资金产生损失的数学期望。P 可表示为 A，B，C 三个等级的预期损失和。

$$P=P_A+P_B+P_C \tag{30}$$

而每个等级企业的的预期损失，为该等级下企业各企业损失期望的和，即：

$$P_{level} = \sum_{i=1}^{num_{level}} Capital_{total} \times (1 - Lose_{In,level_i}) \times r_{level_i} \times Y_{level_i}\ level = A, B, C. \tag{31}$$

3. 确定约束条件

我们确定以下几个约束条件：

$$\textbf{s.t.} \quad \begin{cases} r_A + r_B + r_C = 1 \\ In_{min} \leq In \leq In_{max} \\ PerCapital_{min} \leq Capital_{level_i} \leq PerCapital_{max} \end{cases} \tag{32}$$

其中, 决策变量有 r_{level_i} 和 In($Lose$ 是关于 $level$ 和 In 的函数)。r_{level} 表示银行放贷给 A,B,C 三个等级的企业的金额占年度信贷总额的比率, In 表示银行的贷款年利率。$Capital_{level_i}$ 表示银行给企业 $level_i$ 贷款的额度，$PerCapital_{min}, PerCapital_{max}$ 表示贷款额度的下限（10 万元）和上限（100 万元），In_{min}, In_{max} 表示贷款年利率的下限（4%）和上限（15%）。

$$r_{level} = \sum_{i=1}^{num_{level}} r_{level_i}, level = A, B, C \tag{33}$$

$$Capital_{level_i} = Capital_{total} \times r_{level_i}, i = 1, 2, \cdots, num_{level}, level = A, B, C; \tag{34}$$

5.2 问题二模型的建立

5.2.1 基于 BP 神经网络的四分类模型

不同于问题一，问题二提供的数据了企业的信誉等级，因此无法直接确定某个企业向银行贷款时，其客户流失率与利率的关系，而这个关系在银行的信贷决策中十分重要[7]。

因此，本文利用一个**全连接神经网络**对企业的信誉进行四分类。与问题一的二分类不同的是，我们首先在 11 个二级指标中，筛选出 6 个与企业信誉较为相关的指标，然后对它们做因子分析，标准化、降维到 3 个因子。[8]

该神经网络的输出是一个 4 维向量 $\boldsymbol{L} = (L_A, L_B, L_C, L_D)$. L_{level} 表示该企业信誉等级属于第 $level$ 等级的概率，级该企业对信誉等级 $level$ 的隶属度。针对第 i 家企业，我们可以利用向量 $\boldsymbol{L}^{(i)}$ 预测某企业在利率为 In 时的客户流失率 $PredLose_{i,In}$：

$$PredLose_{i,In} = \sum_{j=0}^{3} L_j^{(i)} \cdot Lose_{In,j} \tag{35}$$

其中 $Lose_{In,j}$ 表示信用等级为 j 的企业在利率 In 下的客户客户流失率。

5.2.2 信贷决策模型

该部分的模型与问题一非常类似，区别在于：问题一的客户流失率为一个确定的、真实的值，而问题二的客户流失率为预测值。故我们只需求出预测的客户流失率，用问题一的方法进行信贷决策即可，在此不再赘述。

[7]详见问题一信贷决策模型建立部分。

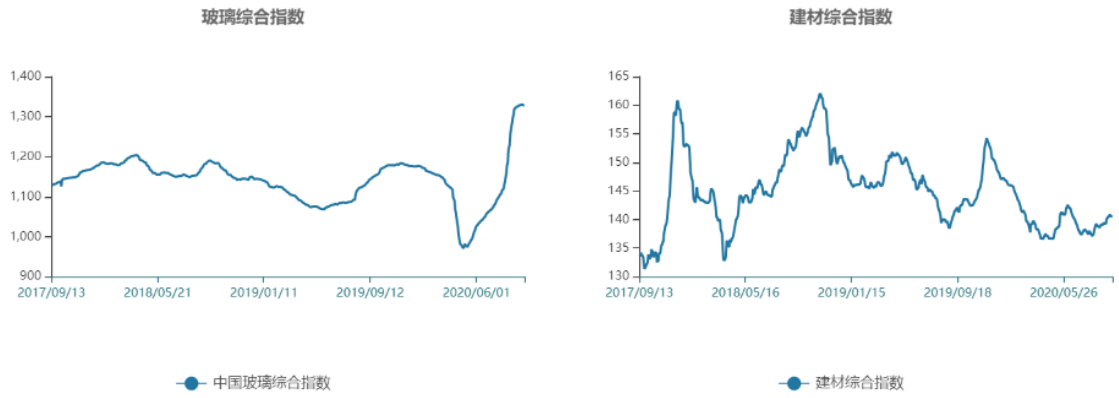
[8]筛选过程和各因子的含义详见问题二的求解部分。

5.3 问题三模型的建立

题目中要求综合考虑突发因素的影响，而突发因素对于不同行业的企业的影响也不尽相同。我们将这些影响映射到信贷风险的计算上，来调整第二问中 1 亿元的分配策略。

5.3.1 风险的划分

以 2020 年新冠肺炎疫情这一突发因素为例，基于相关数据，我们对附录 2 中的企业进行分级：风险敏感型企业[16]，风险稳定型企业和风险受益型企业。风险敏感型企业受风险负面影响大，风险稳定型企业遭遇风险时交易数据变化不显著，风险受益型企业在风险中会获得一定的收益。例如图4中玻璃和建材的综合指数（行业各项指标的综合评估）[9]：



(a) 玻璃综合指数　　(b) 建材综合指数

图 4: 不同行业的企业风险敏感性不同举例：玻璃和建材

为了简化模型，根据新冠疫情实际情况和从 Wind 金融终端分类标准和数据，我们将附件二中第三产业（包括商贸，服务等）和个体经营者定为风险敏感型企业，技术型企业定为风险稳定型企业，一些尤为特殊的包括物流、医疗、游戏类产业定为风险收益型企业。

设突发情况下,风险敏感型企业,风险稳定型企业和风险收益型企业相关业务的损失率分别为 $\alpha_1, \alpha_2, \alpha_3$.

5.3.2 风险影响的指标

突发事件会对净现金流增长率，现金流波动性，销售总额，净税，利润率等方面[2, 15] 产生影响。从这点出发，我们对五个指标进行风险调整:

- 遭遇风险后的净现金流增长率 $=(1-$ 风险损失率$\alpha_i)\times$ 净现金流增长率，$i=1,2,3$
- 遭遇风险后的现金流波动性 $=(1-$ 风险损失率$\alpha_i)\times$ 现金流波动性，$i=1,2,3$
- 遭遇风险后的销售总额 $=(1-$ 风险损失率$\alpha_i)\times$ 销售总额，$i=1,2,3$
- 遭遇风险后的企业净税 $=(1-$ 风险损失率$\alpha_i)\times$ 企业净税，$i=1,2,3$
- 遭遇风险后的企业利润率 $=(1-$ 风险损失率$\alpha_i)\times$ 企业利润率，$i=1,2,3$

进行上述风险调整后，我们可以得到突发事件下，企业各项指标的模拟数据。

[9]数据来源 Wind 金融终端

5.3.3 不同突发事件强度对企业的影响

突发事件的强度不尽相同。我们通过对突发事件强度参数的调整，预测不同企业在不同风险条件下的信贷守约率情况。设突发事件强度参数为 t, 我们通过改变 t 的取值，枚举了在不同风险程度下，银行分配信贷资金的最优策略应当如何制定。

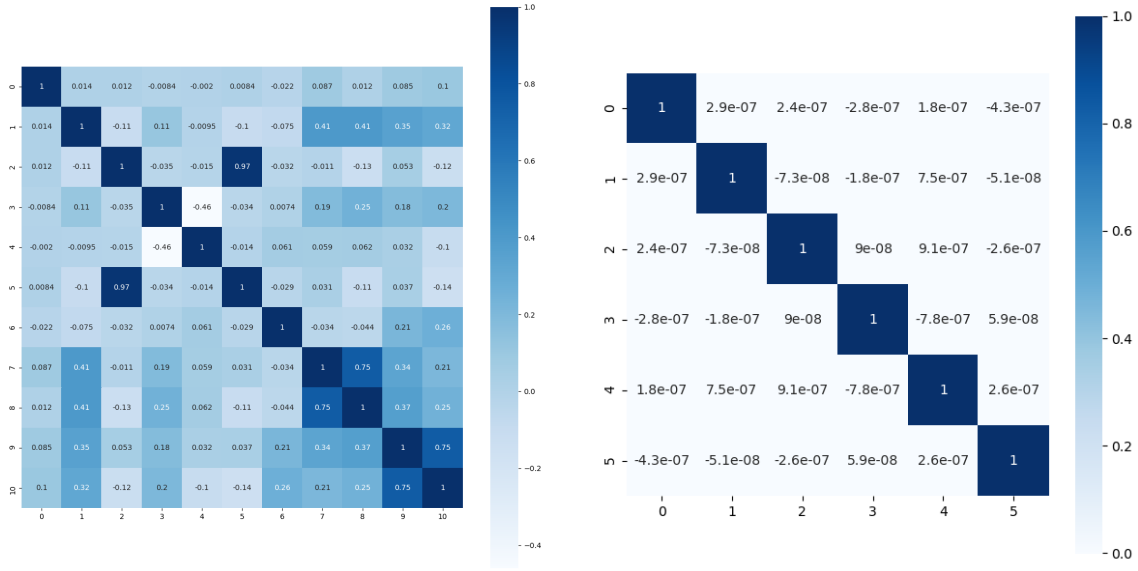
5.3.4 国家政策的影响

突发事件发生时，国家往往会给予的一些政策与经济支持，例如降低税收。《国家税务总局关于调整增值税纳税申报有关事项的公告》（国家税务总局公告 2019 年第 15 号）等文件指出在疫情期间会将一定条件的增值税率降至 9%。在本题中，我们在突发事件发生的前提下，考虑了国家政策中增值税率调整产生的影响，更新预测了各企业的信贷风险，优化了银行的资金配置策略。

6 问题的解答

6.1 问题一的解答

6.1.1 原始指标的因子分析



(a) 因子分析前相关系数图　　(b) 因子分析后相关系数图

图 5: 因子分析前后，各指标 (因子) 相关系数图

各指标的描述性统计见附录表12。可以发现，各个指标的量纲十分不统一，而且根据图5可知，各个指标的相关性较强。因此，我们考虑使用因子分析对数据进行标准化和降维。[14] 表2为因子分析的检验，其中显著性为 0.000，表明因子分析的结果十分显著。KMO 指数大于 0.7 接近 0.8，说明数据十分适合做因子分析。附录中表13为因子分析旋转后的成分矩阵，表中的数值 a_{ij} 表示因子 j 对指标 i 的解释程度。可以看出，每一个因子都有解释程度较高的指标，这表明因子分析的结果准确合理。由附录中表14可见，前 6 个因子解释了 86.091% 的方差，故可认为我们提取的六个因子解释了原先 11 个二级指标中的大部分信息。

图5展示了因子分析前后，各指标（因子）的相关系数对比图。由此可见，经过因子分析提取得到的 6 个因子具有相当好的独立性，并且达到了降维的目的。

表 2: 问题一 KMO 和巴特利特球形度检验

KMO 取样适切性量数		0.793
巴特利特球形度检验	近似卡方	652.186
	自由度	55
	显著性	0.000

6.1.2 二元 Logistic 回归模型求解

在通过因子分析模型求得了 6 个综合因子之后，我们首先将数据集分割为训练集和测试集。具体方法为：将原始数据按照企业名称升序排序后，第 5，10，15 $\cdots$ 个企业作为测试集，其他作为训练集。接着，我们计算了训练集中每个企业 i 相应的因子得分 $(x_{1i}, x_{2i}, \cdots, x_{6i})$。表3给出了回归方程中的估计值 $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_t$, 表4给出了训练集中 Y_i 实测与预测的统计值。可见我们的模型对于原本不违约的企业（$Y_i = 1$）有较好的预测效果（93.6%），而对原本违约的企业（$Y_i = 0$）预测效果较差。总体来看，回归结果对于训练集有 79.4% 的正确率。

将估计值 $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_t$ 代入测试集，计算相应的 Y_i, 结果见附录中的表15，我们的二元 Logistic 回归模型在测试集中的预测正确率为 83.33%。

表 3: 回归方程的中的系数

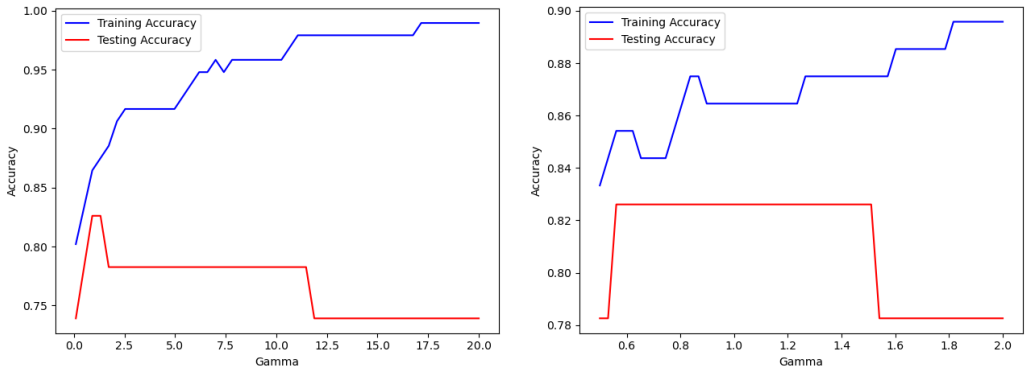
	$\hat{\beta}$	标准误差	瓦尔德	自由度	显著性	Exp(B)
$\hat{\beta}_1$	0.932	1.346	0.480	1	0.489	2.539
$\hat{\beta}_2$	0.778	0.323	5.814	1	0.016	2.178
$\hat{\beta}_3$	0.834	0.331	6.351	1	0.012	2.302
$\hat{\beta}_4$	0.090	0.693	0.017	1	0.897	1.094
$\hat{\beta}_5$	-0.221	0.695	0.101	1	0.750	0.801
$\hat{\beta}_6$	-0.303	0.364	0.691	1	0.406	0.739
$\hat{\beta}_0$	1.880	0.392	23.052	1	0.000	6.553

表 4: 二元 Logistic 回归分类表

实测	预测		
	0	1	正确百分比
0	4	15	21.1
1	5	73	93.6
总体百分比			79.4

6.1.3 基于 SVM 的二分类预测模型求解

分割数据集的方法与逻辑回归相同。我们需要找到一个 γ 值，使得 SVM 在训练集和测试集上的分类精度均在较高水准。图6为训练精度随 γ 变化图。综合分析测试集与训练集的精度变化规律，最终我们选取 $\gamma = 1.0$.



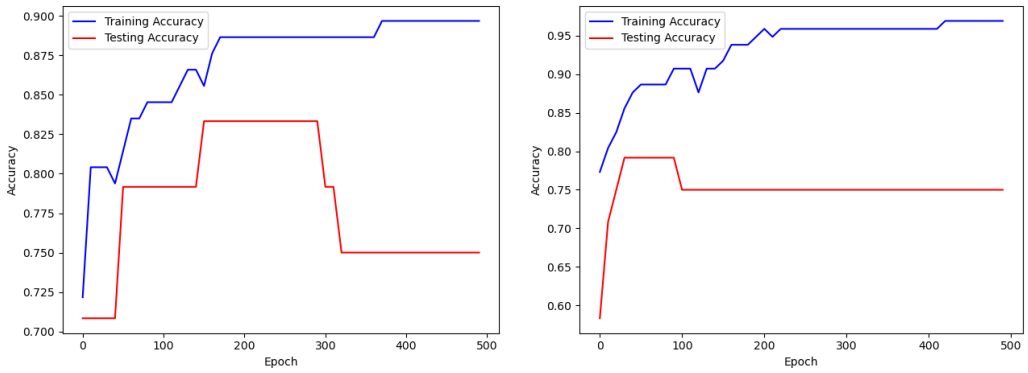
(a) γ 范围较大　　(b) γ 范围较小

图 6: SVM 训练精度随 γ 的变化

经过训练，SVM 在训练集和测试集上的精度均在 0.81 附近，可以用于预测。

6.1.4 基于 BP 神经网络的二分类预测模型求解

神经网络虽然能够取得良好的训练效果，但往往会因为其结构过于庞大或训练世代 (Epoch) 过多导致过拟合[7]。因此，我们采取了与二元逻辑回归相同的分割数据集的办法，将数据集分割为训练集与测试集。同时，将该企业的违约状况作为神经网络的训练目标 (Target)。首先，我们确定了神经网络的结构为全连接神经网络，每层隐藏层的神经元个数为 64，图7为不同隐藏层数目时，训练集和测试集的预测精度随训练代数的变化情况。使用一层隐藏层，在测试集的最高精度可以达到 83.33%，而两层隐藏层虽然在训练集上的训练精度可以高达 95%，但其在测试集的表现不如一层隐藏层，即出现了过拟合的情况。因此，我们确定最终的网络结构为：**一层隐藏层，64 个神经元**。当然，每层中间层均设置了 ReLU 激活函数，最终输出之前会经过 Softmax 激活函数，使得输出结果为一个概率分布。



(a) 一层隐藏层　　(b) 两层隐藏层

图 7: 不同隐藏层个数训练集、测试集的预测精度

如图7所示，一层隐藏层时，测试集的精度从 300 次训练后开始减少，说明 300 代是一个是否过拟合的分界点。在将所有数据均作为训练集时，训练代数需要增加，因此我们选取训练的代数为 300 代。训练完毕后，该模型在训练集上的精度达到了 0.8617，预期在测试集上的精度也将超过 0.8。因此该模型十分适合拿来预测企业是否违约。

6.1.5 二分类模型求解效果比较

针对二分类问题，我们分别使用二元 Logistic 回归、SVM 支持向量机和神经网络三种方法进行训练及预测。表5为不同模型在训练集和测试集上的精度，可以看出：神经网络与二元 Logistic 回归在测试集上表现较好，但神经网络在训练集上的表现更令人满意。因此我们选取神经网络作为二分类的最终模型。

表 5: 二分类模型求解结果

模型名称	训练集精度 (%)	测试集精度 (%)
二元 Logistic 回归	79.40	83.33
SVM 支持向量机	82.98	80.41
全连接 BP 神经网络	89.69	83.33

6.1.6 信贷决策模型求解

通过联立式子(26) - (31)，我们可以对目标函数进行一定量的化简，得到：

$$RAROC_{total} = RAROC_A + RAROC_B + RAROC_C \tag{36}$$

$$RAROC_{level} = \sum_{i=1}^{num_{level}} r_{level_i}(1 - Lose_{In,level_i})(Y_{level_i}(1+In) - (1 - Y_{level_i})) - 1 \tag{37}$$

$$\textbf{s.t.} \quad \begin{cases} r_A + r_B + r_C = 1 \\ In_{min} \le In \le In_{max} \\ PerCapital_{min} \le Capital_{level_i} \le PerCapital_{max} \end{cases} \tag{38}$$

1. 利率的确定

由上式可知，每个企业对于 RAROC 的单位贡献率独立。定义**企业单位回报率**：

$$\lambda_{i,In} = (1 - Lose_{In,level_i}) \cdot (Y_i \cdot (1+In) - (1 - Y_i)) - 1 \tag{39}$$

其中 $\lambda_{i,In}$ 表示银行以利率 In 放贷给企业 i 能获得的单位期望收益。企业将对银行回报的 RAROC 的贡献率，有：

$$RAROC_{level} = \sum_{i=1}^{num_{level}} r_i \cdot \max_{In \in \boldsymbol{Q}} \{\lambda_{In,level_i}\} \tag{40}$$

其中，$\boldsymbol{Q} = (0.04, 0.0425, 0.0465, \cdots, 0.1465, 0.15)$. 对于一个守约率 Y 确定的某 $level$ 企业，对其进行信贷的最佳利率也可以确定。由此我们可以确定对每个贷款企业提供的利率和在风险调整下的银行预期收益率。

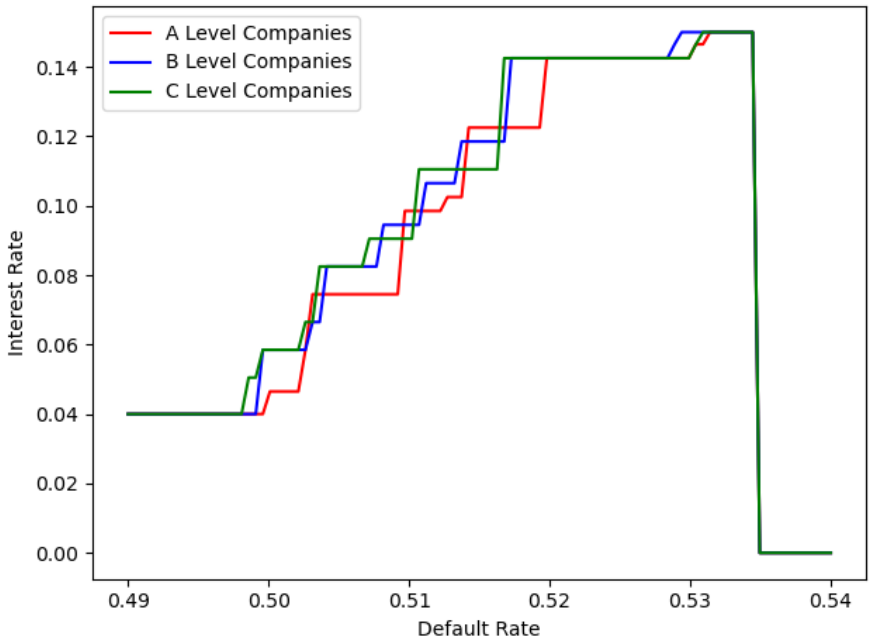


图 8: 不同等级企业最佳利率 In 选择随信贷风险 $1-Y$ 的变化趋势

我们画出了不同等级企业最佳利率 In 选择随信贷风险 $1-Y$ 的变化趋势如图8。对数值结果进行分析，信贷风险 $(1-Y)_{thresh}=0.5345$ 为阈值，大于阈值时银行不予以贷款。风险较低时，三类企业均以利率为 4% 时最优。

2. 银行信贷分配策略

由附件中的数据分析知，企业之间的单位回报率相互独立。我们采用优先放贷给回报率最高的企业的策略。由此生成一个回报率从高到低的序列，即银行给企业放贷的顺序。利用式(39)计算并排序，我们得出分配先后表如表6。

表 6: 借贷企业分配先后表（10 家）

E11	E15	E17	E19	E20	E23	E24	E26	E27	E31

下面我们将对表6中的 10 家企业进行借贷分配，并对固定信贷总额 $Capital_{total}$ 进行分类讨论。定义下述指标（单位：万元）：

$$Number = \lfloor \text{银行年度信贷总额} Capital_{total}/100 \rfloor \tag{41}$$

即

$$Capital_{total} = 100 \cdot Number + Rest \tag{42}$$

方案一 若 $Rest \in [10, 90]$，采取策略为，给序列中前 $Number$ 家企业分别分配 100 万元，将 $Rest$ 的资金分配给 $Number+1$ 家企业，以此获得最大收益。

方案二 若 $Rest \in [0, 10]$，此时银行面临两个选择：

A 给序列中前 $Number$ 家企业分别分配 100 万元，将 $Rest$ 的资金闲置（因为银行有 10 万元放贷下限）。

B 给序列中前 $Number-1$ 家企业分别分配 100 万元，第 $Number$ 家分配 $90+Rest$ 万元，第 $Number+1$ 家分配 10 万元。

分别求两种策略的 $RAROC_{total}$, 作差后可得:

- 若 $Y_{Number+1} < \frac{-2Y_{Number} \cdot Rest + 20Y_{Number} + Rest}{20}$, 采取方案二.A
- 若 $Y_{Number+1} = \frac{-2Y_{Number} \cdot Rest + 20Y_{Number} + Rest}{20}$, 方案二中 A,B 方案效果一致
- 若 $Y_{Number+1} > \frac{-2Y_{Number} \cdot Rest + 20Y_{Number} + Rest}{20}$, 采取方案二.B

由于守约率 $Y_{Number+1}$ 和 Y_{Number} 已知，可以通过关系式确定银行放贷的方案。

6.2 问题二的解答

6.2.1 四分类的指标筛选

在问题一中，我们确立了 4 个一级指标与 11 个二级指标，如图2所示。其中部分指标与企业的信誉相关性较低，我们考虑将其剔除。指标与信誉的相关性判断参考了王月萍的判别方法[10]。

首先，净现金流增长率与企业的信誉相关性较低。这个指标由一个企业的成长能力与其所处行业、客观经济条件和企业战略等因素决定，与企业信誉不够相关。

其次，净现金流入流出比和利润率反应的是企业财务上的状况，信誉好的企业也会存在亏损情况，而信誉不好的企业也可能盈利颇丰，因此予以剔除。

此外，企业的上游供应稳定性的两个指标也与企业的信誉关系不密切。因为一个企业更换供应商的理由有很多，有可能是厂家质量问题，也有可能是厂家的违约造成的。

剩下的指标有：**现金流波动性**，波动较小的企业从时间维度上看较为稳定，也往往拥有较高的信誉；**销售总额**和**净税**，反映了企业的规模，一般来说规模大的企业信誉高；**退货比例**，反映企业的诚信，退货比例过高很可能是由于企业自身问题造成的；**下游需求稳定性指标 1、2**，反映企业的客户稳定度，一般来说，拥有稳定客户群的企业拥有较高的信誉。

表7为筛选后的指标进行因子分析前的 KMO 和巴特利特球形度检验，其中显著性为 0.000，表示结果显著，KMO 指数大于 0.7 表示数据适合因子分析。

表 7: 附件一四分类 KMO 和巴特利特球形度检验

KMO 取样适切性量数		0.772
巴特利特球形度检验	近似卡方	165.844
	自由度	15
	显著性	0.000

6.2.2 基于 BP 神经网络的四分类模型求解

确定因子后，我们将附件一进行因子分析后的数据集划分为训练集与测试集，并测试了在不同的网络结构下，预测精度与训练代数的关系，最终选择的结构为：**一层隐藏层，128 个神经元**。[10]同时，为了测试式(35)的准确性，我们对测试集的客户流失率进行了验证，均方误差 (MSE) 为 0.04，结果较好，进而验证了预测方法的合理性。[11]

[10]选择的方法类似问题一，不再赘述。

[11]计算代码详见支撑材料 `Loss_rate.py`，预测结果详见支撑材料 `pred_loss.xls`.

得出了客户流失率后，结合之前得出的信贷风险，我们可以带入信贷决策模型求解期望收益最大时的策略，如伪代码1所示，其中 N_{rates} 表示利率等级的个数，$rate_i$ 表示第 i 个利率等级的真实利率。具体的信贷策略见表8，表中结果为不同企业的优先级顺序。

Algorithm 1: 问题二信贷决策模型求解

Input: 第 i 家企业的守约率 Y_i，利率为 j 时的客户流失率 $PredLoss_{i,j}$.

Output: 银行贷款给第 i 家企业的资金比例 r_i，利率 In_i.

初始化 $ans \leftarrow 0$, $\boldsymbol{r} \leftarrow \mathbf{0}$, $\boldsymbol{In} \leftarrow \mathbf{0}$;

for $i \leftarrow 1$ **to** N_{rates} **do**

  在利率 $rate_i$ 下，求解式(36)描述的线性规划问题;

  求解结果的目标函数为 $objective$, 最优解为 $\boldsymbol{x}$;

  if 当前目标函数最优值 $objective > ans$ **then**

    $ans \leftarrow objective$;

    $\boldsymbol{r} \leftarrow \boldsymbol{x}$;

    $In_i \leftarrow rate_i$;

  end

end

表 8: 问题二最优解序列

E136	E142	E145	E152	E154	E155	E158	E163	E167	E181
E186	E187	E188	E216	E217	E225	E229	E231	E235	E241
E243	E247	E258	E261	E264	E270	E277	E281	E282	E283
E284	E287	E289	E293	E297	E298	E304	E315	E318	E324
E325	E326	E327	E329	E330	E332	E333	E334	E342	E352
E357	E360	E369	E379	E387	E388	E390	E395	E398	E403
E409	E419	E421	E180	E220	E242	E269	E280	E288	E299
E313	E335	E343	E345	E346	E373	E410	E244	E256	E420
E321	E194	E291	E301	E185	E173	E153	E183	E249	E175
E290	E148	E402	E273	E309	E314	E384	E361	E202	E128

[1] 优先级顺序：从左到右，从上到下，下同。本例中，E136 最优，其次 E142. 第二问决策为向以上企业放贷 100 万元。

6.3 问题三的解答

6.3.1 风险的划分

我们通过疫情数据，分析突发事件对企业信贷产生影响。利用《国盛证券-宏观点评：央行四季度货币政策报告的 7 大信号》数据表中的数据（如图9）。对疫情期间的各行业企业的风险损失率 $\alpha_1, \alpha_2, \alpha_3$ 进行测算：

$$\alpha_i = \frac{\alpha_{i,\text{乐观}} + 4 \cdot \alpha_{i,\text{中立}} + \alpha_{i,\text{悲观}}}{6}, \quad i = 1, 2, 3 \tag{43}$$

近似求得：风险损失率 $\alpha_1, \alpha_2, \alpha_3$ 分别为 $0.3, 0.05, -0.2$

结果汇总(百分点)		第二产业	工业	建筑业	第三产业	批发和零售业	交通运输、仓储和邮政业	住宿和餐饮业	金融业	房地产业	信息传输、软件和信息技术服务业	租赁和商务服务业	其他行业
非典时期冲击回顾													
非典时期变动（03年Q2-Q1）		-1.90	-2.00	-1.70	-1.80	2.00	-5.40	-3.60	-3.60	1.60	—	—	-2.20
对03年Q2GDP拖累		-0.89	-0.82	-0.10	-0.77	0.17	-0.32	-0.08	-0.16	0.07	—	—	-0.37
对03年GDP的拖累		-0.22	-0.21	-0.02	-0.19	0.04	-0.08	-0.02	-0.04	0.02	—	—	-0.09
新冠肺炎冲击预测													
对2020年Q1当季拖累（预测）	乐观	-0.11	-0.10	-0.02	-0.98	-0.20	-0.36	-0.07	0.00	-0.07	-0.03	-0.03	-0.22
	中观	-0.26	-0.23	-0.04	-1.56	-0.30	-0.40	-0.09	-0.09	-0.15	-0.07	-0.06	-0.40
	悲观	-0.38	-0.33	-0.05	-2.14	-0.40	-0.45	-0.11	-0.18	-0.22	-0.11	-0.10	-0.58
对2020年Q2当季拖累（预测）	乐观	-0.32	-0.26	-0.06	-0.22	0.00	-0.24	-0.04	0.08	0.00	0.01	0.01	-0.03
	中观	-0.48	-0.39	-0.09	-0.76	-0.10	-0.29	-0.06	0.00	-0.07	-0.03	-0.03	-0.19
	悲观	-0.64	-0.53	-0.12	-1.29	-0.19	-0.33	-0.08	-0.08	-0.14	-0.06	-0.06	-0.35
2020年Q1对全年的拖累（假设疫情持续至一季度全年GDP可能的变动）	乐观	-0.02	-0.02	0.00	-0.22	-0.04	-0.08	-0.01	0.00	-0.02	-0.01	-0.01	-0.05
	中观	-0.06	-0.05	-0.01	-0.34	-0.07	-0.09	-0.02	-0.02	-0.03	-0.02	-0.01	-0.09
	悲观	-0.08	-0.07	-0.01	-0.47	-0.09	-0.10	-0.02	-0.04	-0.05	-0.02	-0.02	-0.13
2020年Q2对全年的拖累	乐观	-0.08	-0.07	-0.01	-0.06	0.00	-0.06	-0.01	0.02	0.00	0.00	0.00	-0.01
	中观	-0.12	-0.10	-0.02	-0.19	-0.02	-0.07	-0.01	0.00	-0.02	-0.01	-0.01	-0.05
	悲观	-0.16	-0.13	-0.03	-0.32	-0.05	-0.08	-0.02	-0.02	-0.04	-0.02	-0.01	-0.09
2020年Q1+Q2对全年的拖累（假设疫情持续至二季度全年GDP可能的变动）	乐观	-0.11	-0.09	-0.02	-0.27	-0.04	-0.14	-0.03	0.02	-0.02	-0.01	0.00	-0.06
	中观	-0.18	-0.15	-0.03	-0.53	-0.09	-0.16	-0.03	-0.02	-0.05	-0.02	-0.02	-0.14
	悲观	-0.24	-0.20	-0.04	-0.79	-0.14	-0.18	-0.04	-0.06	-0.08	-0.04	-0.04	-0.22

结果概要(百分点)		第二产业	第三产业	二三产业合计
非典时期冲击回顾				
非典时期变动（03年Q2-Q1）		-1.90	-1.80	—
对03年Q2GDP拖累		-0.89	-0.77	-1.7
对03年GDP的拖累		-0.22	-0.19	-0.4
新冠肺炎冲击预测				
对2020年Q1当季拖累（预测）	乐观	-0.11	-0.98	-1.1
	中观	-0.26	-1.56	-1.8
	悲观	-0.38	-2.14	-2.5
对2020年Q2当季拖累（预测）	乐观	-0.32	-0.22	-0.5
	中观	-0.48	-0.76	-1.2
	悲观	-0.64	-1.29	-1.9
2020年Q1对全年的拖累（假设疫情持续至一季度全年GDP可能的变动）	乐观	-0.02	-0.22	-0.2
	中观	-0.06	-0.34	-0.4
	悲观	-0.08	-0.47	-0.6
2020年Q2对全年的拖累	乐观	-0.08	-0.06	-0.1
	中观	-0.12	-0.19	-0.3
	悲观	-0.16	-0.32	-0.5
020年Q1+Q2对全年的拖累（假设疫情持续至二季度全年GDP可能的变动）	乐观	-0.11	-0.27	-0.4
	中观	-0.18	-0.53	-0.7
	悲观	-0.24	-0.79	-1.0

图 9: 风险划分

6.3.2 风险对企业指标的影响

通过风险损失率更新相应的业务指标。即

$$X_j^{'} = (1 - \alpha_i) \cdot X_j \quad i = 1, 2, 3 \tag{44}$$

将 $X_j^{'}$ 重新进行因子分析，二分类预测和信贷决策，表9列举出了突发情况下银行 1 亿元信贷的 100 家企业，信贷金额为 100 万元。其中不难看出，信贷顺序位次有了巨大变化。$E401, E361, E295, E406, E352, E316$ 成为信贷回报率位次前五的企业。

6.3.3 不同风险程度对企业的影响

我们引入参数 t 来模拟突发事件强度的不同。

$$t = 0.5, 0.7, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4, 1.5$$

其中 $t > 1$ 表示强度大于此次疫情，$t < 1$ 表示强度小于此次疫情。优化后的指标如下：

$$X_i^{'} = t \cdot (1 - \alpha_1)_i \tag{45}$$

在参数 $t = 1.0$ 的情况下，银行的最优决策如表9所示。

表 9: 突发事件强度参数 $t = 1.0$ 时，求得的贷款优先级

E401	E361	E295	E406	E352	E316	E244	E343	E366	E327
E357	E395	E379	E410	E222	E256	E321	E299	E392	E285
E153	E373	E414	E388	E298	E266	E409	E281	E309	E369
E330	E365	E408	E413	E348	E384	E312	E346	E387	E229
E394	E347	E264	E390	E398	E270	E183	E420	E258	E336
E338	E273	E287	E412	E277	E424	E305	E280	E399	E421
E279	E325	E422	E247	E188	E385	E283	E207	E235	E294
E145	E291	E417	E423	E301	E403	E252	E250	E360	E375
E225	E335	E349	E364	E186	E282	E419	E345	E318	E259
E177	E350	E354	E227	E261	E174	E220	E329	E288	E313

参数 t 取其他值的最优决策的企业优先级对照表如表10所示。

表 10: 不同参数 t 下的最优决策企业优先级对照表

突发事件强度参数 t	企业优先级表格
0.5	见附录表25
0.7	见附录表26
0.8	见附录表27
0.9	见附录表28
1.1	见附录表29
1.2	见附录表30
1.3	见附录表31
1.4	见附录表32
1.5	见附录表33

6.3.4 国家政策影响

在新冠疫情期间，国家为支持国民经济发展，施行降低增值税等政策[12]。我们在突发事件的基础上，模拟了税收的减免，定义 t_{tax} 为税率减免率。我们将税率单独处理:

$$X_{tax}^{''} = (1 - t_{tax}) \cdot t \cdot (1 - alpha_1) \cdot X_i \tag{46}$$

表 11: 税额减免率 $t_{tax} = 0.5$ 时，银行的最优解策略

E216	E186	E365	E241	E278	E359	E193	E342	E348	E324
E312	E315	E333	E260	E289	E369	E282	E243	E153	E313
E402	E341	E334	E373	E262	E284	E314	E269	E378	E180
E304	E290	E292	E287	E299	E242	E167	E388	E387	E306
E148	E326	E182	E249	E298	E254	E329	E394	E288	E139
E178	E211	E297	E409	E246	E413	E332	E325	E240	E293
E283	E301	E185	E280	E392	E141	E281	E408	E305	E384
E154	E346	E152	E195	E390	E127	E187	E398	E420	E424
E285	E399	E253	E412	E270	E175	E291	E422	E300	E379
E361	E421	E401	E220	E181	E172	E417	E210	E303	E256

将 t_{tax} 分别设定为 $0.5, 0.6, 0.7, 0.8, 0.9$. 对应的最优解序列详见附录34-37。注意到，随着税率的下降，小微企业获得信贷的概率不断攀升了。

7 灵敏度分析

鉴于本题的特殊性，问题三的求解过程实质上就是模型的灵敏度分析。我们对环境模拟了不同的意外状况（改变突发事件强度参数 t），求得了不同突发事件强度的情况下（包括对企业有正向影响和负向影响的突发事件），银行借贷的最优解序列，对应附录表25 - 33。

[12]参考 http://www.chinatax.gov.cn/chinatax/n810219/n810780/c5148506/content.html

此外，我们还考虑了政府应对突发情况所采取的措施（主要是对企业减税）对银行信贷决策的影响（参数 t_{tax}），最优解序列对应附录表11 - 37。

综合分析最优解序列，我们可以得出以下结论：本文所建立的银行信贷决策模型鲁棒性较强。这是由于：虽然决策出的最优解序列中，企业顺序有所变动，但针对序列中的每一个企业，银行放出的贷款数额均接近 100 万元；并且，有多数企业[13]在突发事件强度参数 t 改变的过程中，稳定地出现在了最优解序列里。

上述结果其实不光证明我们所建立的信贷决策模型具有良好的稳定性，同时也证明了二分类模型以及四分类模型具有较强的鲁棒性。这是因为银行信贷决策时基于二分类模型和四分类模型的结果，而一般情况下，稳定的输出具有较为稳定的输入[3]，且通过对二分类模型输出 Y_i 和四分类模型输出 $\boldsymbol{L}^{(i)}$ 的综合分析，发现它们的变动较小[14]。因此本文建立的分类模型鲁棒性强。

8 模型总结

8.1 模型优点

1. 各项指标的设立具有坚实的经济学理论作为支撑：
   (a) 结合题目所给数据，巧妙地建立了基于准确的现金流评价体系；
   (b) 从企业现金流状况、企业规模、销售质量及供求关系稳定性四个不同的方面综合评判企业实力，指标丰富、评价方式合理；
2. 因子分析的使用恰到好处，既消除了指标间量纲的差距，又显著地降低了相关系数；
3. 对比了二分类预测模型求解的各种方法，择优采用；
4. 提出隶属度的概念，增强了四分类预测模型的可解释性，也显著提高了客户流失率的预测精度；

8.2 模型缺点

1. 模型数据来源较为单一，主要数据仅为企业的发票流水，缺失企业的负债数据，从而对企业偿债能力的评价不够精准；
2. 对企业行业的区分不够细致，仅依靠企业名称中的关键词来判别企业所处行业，且名称为“个体经营”的企业难以判断其所处行业，从而导致问题三的突发事件量化程度不够高；
3. 二元 Logistic 回归预测模型，对原本违约的企业预测不够准确；
4. 尚未考虑数据增强的手段，例如使用对抗神经网络 (GAN) 生成训练数据。

9 展望

本模型在未来有多种优化方式，如：

1. 更改神经网络结构，提高训练效率及精度；

[13]如企业 E128, E136, E149, E163 等。

[14]对于突发事件时 $Y_i > 0.99$ 的企业，无论外界怎么变动，其 Y_i 总是大于 0.9.

2. 收集更全面的数据，如企业的行业信息及其负债信息，建立更加全面的指标；

3. 增加数据样本量，使得二分类及四分类模型在避免过拟合的前提下具有更高的拟合精度。

参考文献

[1] Dermine J.P. It falls in the application of raroc in loan management. *The Arbitrageur,1998(3):1-3.*

[2] 于长雷 and 赵红. 重大网络突发事件对房地产市场的影响研究. 管理评论, 28(008):66–70, 2016.

[3] 吴旭东 and 解学书. H∞ 鲁棒控制中的加权阵选择. 清华大学学报 *(自然科学版)*, (1):27–30, 1997.

[4] 尹帅. 基于可持续发展的首都机场财务评价体系研究. PhD thesis, 北京交通大学.

[5] 张玉. 小微企业与银行信贷关系的博弈策略探析. 金融监管研究, (04):94–106, 2012.

[6] 施炳展 and 邵文波. 中国企业出口产品质量测算及其决定因素——培育出口竞争新优势的微观视角. 管理世界, 000(009):90–106, 2014.

[7] 李俭川, 秦国军, 温熙森, and 胡茑庆. 神经网络学习算法的过拟合问题及解决方法. 振动, 测试与诊断, (04):16–20+76, 2002.

[8] 李秉成, 田笑丰, and 曹芳. 现金流量表分析指标体系研究. 福建金融, 000(010):25–29, 2003.

[9] 王明瑞. 基于现金流的高速公路企业财务评价研究. PhD thesis, 2015.

[10] 王月萍. 企业信誉的多层次灰色评估模型. 山西大学学报 *(自然科学版)*, (04):463–467, 2007.

[11] 白春鹤. 广西林下经济供求关系和 *SWOT* 决策研究. PhD thesis, 2017.

[12] 聂辉华, 谭松涛, and 王宇锋. 创新、企业规模和市场竞争: 基于中国企业层面的面板数据分析. 世界经济, 031(007):57–66, 2008.

[13] 胡诗君. 现金流波动性对企业价值影响的实证研究. PhD thesis, 浙江大学, 2015.

[14] 苏为华. 多指标综合评价理论与方法问题研究. PhD thesis, 厦门大学, 2000.

[15] 覃艳华. 突发事件下多因素同时扰动的闭环供应链协调策略. 数学的实践与认识, (06):109–119, 2015.

[16] 陈国栋. 基于蒙特卡罗模拟的投资项目风险敏感性分析. 财会月刊, 000(006):59–61, 2012.

[17] 陈晓光. 增值税有效税率差异与效率损失——兼议对”营改增”的启示. 中国社会科学, 000(008):67–84, 2013.

附录

A 重要代码

部分 Python 程序关联到文件路径，如果要试运行，需要更改为正确的路径。

A.1 数据预处理代码

净现金流增长率、现金流波动性、现金流入流出比、销售总额、净税及利润率的处理仅需要依靠 EXCEL 的数据透视表功能即可完成。

退货比例处理代码：tuihuo.py

输入：用数据透视表处理出的表格，其格式与支撑材料中退货（处理前）.xlsx 相同。

输出：outputs[i]，表示第 i 家企业的退货金额占比。

```
import os
import xlrd
import xlwt
import numpy as np

NAME ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\问题二\\退货(处理前).xlsx"
NUMBER =302

if __name__ =="__main__":
    data =xlrd.open_workbook(NAME)
    table =data.sheet_by_name('Sheet1')
    N =table.nrows

    # outputs[i, 0] 表示第 i 家企业两年回头客消费占的资金比例
    # outputs[i, 1] 表示第 i 家企业三年回头客消费占的资金比例
    outputs =np.zeros((NUMBER,))

    # 处理进项
    i =0
    while i <N:
        if table.cell(i, 0).value[0] =="E":
            name =int(table.cell(i, 0).value[1:]) -124
            total =float(table.cell(i, 1).value)
            two_year =0
            three_year =0

            # find next company
            tail =i
            while (tail +1 <=N -1) and (table.cell(tail +1, 0).value ==table.cell(tail, 0).value):
                tail +=1

            # i 到 tail 的数据是有效的
            # tail + 1 是下一个公司
            for k in range(i, tail):
                outputs[name] +=table.cell(k, 1).value

```

```
        i = tail +1

    wb = xlwt.Workbook()
    sheet1 = wb.add_sheet("Sheet1")
    for i in range(NUMBER):
        sheet1.write(i, 0, outputs[i])
    wb.save('In and Out.xls')
```

上游企业稳定性处理代码：Stability_jin.py

输入：用数据透视表处理出的表格，其形式与指证材料中不同公司的上游企业（处理前).xlsx 相同。

输出：outputs[i, 0] 和 outputs[i, 1] 分别表示第 i 家企业上游供求稳定性指标 1, 2.

```
import os
import xlrd
import xlwt
import numpy as np

NAME ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\问题二\\上游企业(处理前).xlsx"
NUMBER =302

if __name__ =="__main__":
    data =xlrd.open_workbook(NAME)
    table =data.sheet_by_name('Sheet1')
    N = table.nrows

    # outputs[i, 0] 表示第 i 家企业两年回头客消费占的资金比例
    # outputs[i, 1] 表示第 i 家企业三年回头客消费占的资金比例
    outputs =np.zeros((NUMBER, 2))

    # 处理进项
    i = 0
    while i <N:
        if table.cell(i, 0).value[0] =="E":
            name =int(table.cell(i, 0).value[1:]) -124
            total =float(table.cell(i, 1).value)
            two_year =0
            three_year =0

            # find next company
            tail =i
            while (tail +1 <=N -1) and (table.cell(tail +1, 0).value[0] !="E"):
                tail +=1

            # i + 1 到 tail 的数据是有效的
            # tail + 1 是下一个公司
            start =i +1
            while start <tail +1:
                # Begin to processing
                if table.cell(start, 0).value[0] =="C":
                    end =start
                    while (end +1 <=N -1) and (table.cell(end +1, 0).value[0] !="C") and (table.cell(end
```

```
                                                    +1, 0).value[0] !="E"):
                    end +=1

                # start 到 end 的数据是有效的
                # end + 1 是下一个销地
                if end -start >=2:
                    two_year +=float(table.cell(start, 1).value)

                if end -start >=3:
                    three_year +=float(table.cell(start, 1).value)

                start =end +1

        if total ==0:
            outputs[name, 0] =0
            outputs[name, 1] =0
        else:
            outputs[name, 0] =two_year /total
            outputs[name, 1] =three_year /total

        i =tail +1

wb =xlwt.Workbook()
sheet1 =wb.add_sheet("Sheet1")
for i in range(NUMBER):
    sheet1.write(i, 0, outputs[i, 0])
    sheet1.write(i, 1, outputs[i, 1])
wb.save('In and Out.xls')
```

上游企业稳定性处理代码：Stability_xiao.py

输入：用数据透视表处理出的表格，其形式与指证材料中不同公司的下游企业（处理前).xlsx 相同。

输出：outputs[i, 0] 和 outputs[i, 1] 分别表示第 i 家企业下游需求稳定性指标 1, 2.

```
import os
import xlrd
import xlwt
import numpy as np

NAME ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\问题二\\下游企业(处理前).xlsx"
NUMBER =302

if __name__ =="__main__":
    data =xlrd.open_workbook(NAME)
    table =data.sheet_by_name('Sheet1')
    N =table.nrows

    # outputs[i, 0] 表示第 i 家企业两年回头客消费占的资金比例
    # outputs[i, 1] 表示第 i 家企业三年回头客消费占的资金比例
    outputs =np.zeros((NUMBER, 2))

    # 处理进项
```

```
    i = 0
    while i <N:
        if table.cell(i, 0).value[0] =="E":
            name =int(table.cell(i, 0).value[1:]) -124
            total =float(table.cell(i, 1).value)
            two_year =0
            three_year =0

            # find next company
            tail =i
            while (tail +1 <=N -1) and (table.cell(tail +1, 0).value[0] !="E"):
                tail +=1

            # i + 1 到 tail 的数据是有效的
            # tail + 1 是下一个公司
            start =i +1
            while start <tail +1:
                # Begin to processing
                if table.cell(start, 0).value[0] =="D":
                    end =start
                    while (end +1 <=N -1) and (table.cell(end +1, 0).value[0] !="D") and (table.cell(end
                                                                                        +1, 0).value[0] !="E"):
                        end +=1

                    # start 到 end 的数据是有效的
                    # end + 1 是下一个销地
                    if end -start >=2:
                        two_year +=float(table.cell(start, 1).value)

                    if end -start >=3:
                        three_year +=float(table.cell(start, 1).value)

                    start =end +1

            outputs[name, 0] =two_year /max(1, total)
            outputs[name, 1] =three_year /max(1, total)

            i =tail +1

    wb =xlwt.Workbook()
    sheet1 =wb.add_sheet("Sheet1")
    for i in range(NUMBER):
        sheet1.write(i, 0, outputs[i, 0])
        sheet1.write(i, 1, outputs[i, 1])
    wb.save('In and Out.xls')
```

A.2 二分类代码

A.2.1 训练代码

在运行代码之前，请确保输入表格的形式与支撑材料总数据**.xlsx 相一致：**第一列第二行开始为企业标号，第二列为是否违约，后面每一列为一个指标。

1. SVM 代码：SVM.py

输入：形式如总数据.xlsx 中名为因子 6 的表单的因子分析表。

输出：不同 γ 下训练集和测试集的精度的变化曲线。

```
from sklearn import svm
import pandas as pd
import numpy as np
import xlrd
import matplotlib.pyplot as plt

data_train =pd.read_excel('train and test.xls', sheet_name=0)
data_test =pd.read_excel('train and test.xls', sheet_name=1)

x_train =data_train.iloc[:, 1:]
y_train =data_train.iloc[:, 0]
x_test =data_test.iloc[:, 1:]
y_test =data_test.iloc[:, 0]

npx_train =np.array(x_train)
npy_train =np.array(y_train)
npx_test =np.array(x_test)
npy_test =np.array(y_test)

gammas =np.linspace(0.5, 2)
train_acc =[]
test_acc =[]
for gamma in gammas:
    clf =svm.SVC(C=1., kernel='rbf', gamma=gamma, decision_function_shape='ovr')
    clf.fit(npx_train, npy_train.ravel())
    y_hat =clf.predict(npx_train)
    train_acc.append(sum((y_hat ==npy_train) *1) /npx_train.shape[0])
    # print(y_hat)
    # print("Accuracy:", sum((y_hat == npy_train) * 1) / npx_train.shape[0])

    y_hat =clf.predict(npx_test)
    test_acc.append(sum((y_hat ==npy_test) *1) /npx_test.shape[0])
    # print(y_hat)
    # print("Accuracy:", sum((y_hat == npy_test) * 1) / npx_test.shape[0])

plt.plot(gammas, train_acc, color='blue', label='Training Accuracy')
plt.plot(gammas, test_acc, color='red', label='Testing Accuracy')
plt.xlabel('Gamma')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

2. 二分类神经网络验证代码： Classification_validation.py

输入：同 SVM.py.

输出：训练集和测试集精度随训练代数增加的变化曲线图。

```
import os
import xlrd
import xlwt
import torch
import datetime
import numpy as np
import torch.nn as nn
import torch.utils.data as Data
import matplotlib.pyplot as plt

torch.manual_seed(1)

PATH ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\总数据.xlsx"
BATCH_SIZE =8
EPOCH =500

class RNN(nn.Module):
    def __init__(self):
        super(RNN, self).__init__()
        self.rnn =nn.LSTM(
            input_size=input_size,
            hidden_size=64,
            num_layers=2,
            batch_first=True,
        )
        self.out =nn.Linear(64, 2)

    def forward(self, x):
        r_out, (h_n, h_c) =self.rnn(x, None)

        out =self.out(r_out[:, -1, :])
        return out

if __name__ =="__main__":
    # load data
    data =xlrd.open_workbook(PATH)
    table =data.sheet_by_name("因子6")
    input_number =table.nrows -1
    input_size =table.ncols -2

    train_size =input_number -input_number //5
    train_x =torch.zeros((train_size, input_size))
    train_y =torch.zeros((train_size,))
    test_x =torch.zeros((input_number -train_size, input_size))
    test_y =torch.zeros((input_number -train_size,))

    # transform from excel file to torch tensor
```

```
    train_p =0
    test_p =0
    for i in range(1, table.nrows):
        if i % 5 !=0:
            for j in range(input_size):
                train_x[train_p, j] =table.cell(i, j +2).value
            train_y[train_p] =table.cell(i, 1).value
            train_p +=1
        else:
            for j in range(input_size):
                test_x[test_p, j] =table.cell(i, j +2).value
            test_y[test_p] =table.cell(i, 1).value
            test_p +=1

    # normalize
    '''
    for j in range(input_size):
        mean = train_x[:, j].mean()
        std = train_x[:, j].std()
        train_x[:, j] = (train_x[:, j] - mean) / std
        # train_x[:, j] = train_x[:, j] / mean

        mean = test_x[:, j].mean()
        std = test_x[:, j].std()
        test_x[:, j] = (test_x[:, j] - mean) / std
        # test_x[:, j] = test_x[:, j] / mean
    '''
    '''
    wb = xlwt.Workbook()
    sheet1 = wb.add_sheet("train")
    sheet2 = wb.add_sheet("test")
    for i in range(train_size):
        for j in range(input_size):
            sheet1.write(i, j+1, train_x[i, j].item())
        sheet1.write(i, 0, train_y[i].item())
    for i in range(input_number - train_size):
        for j in range(input_size):
            sheet2.write(i, j+1, test_x[i, j].item())
        sheet2.write(i, 0, test_y[i].item())
    wb.save('train and test.xls')
    '''

    # create data loader
    dataset =Data.TensorDataset(train_x, train_y)
    data_loader =Data.DataLoader(
        dataset=dataset,
        batch_size=BATCH_SIZE,
        shuffle=True,
    )

    device =torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
model =nn.Sequential(
    nn.Linear(input_size, 64),
    nn.ReLU(),
    nn.Linear(64, 2),
    nn.Softmax(),
).to(device)
# model = RNN().to(device)

optimizer =torch.optim.Adam(model.parameters(), lr=1e-3)
loss_func =torch.nn.CrossEntropyLoss()

train_ACC =[]
test_ACC =[]
in_X =[]

for epoch in range(EPOCH):
    loss_all =[]
    for step, (batch_x, batch_y) in enumerate(data_loader):
        x = batch_x.type(torch.FloatTensor).to(device)
        y = batch_y.type(torch.LongTensor).to(device)
        # pred = model.forward(x.view(-1, 1, input_size)) # (batch, time_step, input_size)
        pred =model.forward(x)
        loss =loss_func(pred, y)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        loss_all.append(loss.cpu().item())

    if epoch % 10 ==0:
        # calculate accuracy on testing data
        # pred = model.forward(test_x.view(-1, 1, input_size).to(device))
        pred =model.forward(test_x.to(device))
        pred_y =torch.max(pred, 1)[1].cpu().data.numpy()
        target_y =test_y.data.numpy()
        accuracy =float((pred_y ==target_y).astype(int).sum()) /float(target_y.size)

        # pred = model.forward(train_x.view(-1, 1, input_size).to(device))
        pred =model.forward(train_x.to(device))
        pred_y =torch.max(pred, 1)[1].cpu().data.numpy()
        target_y =train_y.data.numpy()
        train_acc =float((pred_y ==target_y).astype(int).sum()) /float(target_y.size)
        print("|| Epoch {} | Loss {:.4f} | train_acc {:.4f} | test_acc {:.4f} ||".format(epoch, np.
                                                                    mean(loss_all),
                                                                                train_acc, accuracy))

        in_X.append(epoch)
        train_ACC.append(train_acc)
        test_ACC.append(accuracy)

```

```
    plt.plot(in_X, train_ACC, color='blue', label='Training Accuracy')
    plt.plot(in_X, test_ACC, color='red', label='Testing Accuracy')
    plt.legend()
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.show()
```

3. 二分类神经网络训练代码： Classification_train.py

输入：同 Classification_validation.py.

输出：以.h5 形式存储的 PyTorch 模型参数，保存在 models 文件夹下。模型文件名含义为训练完成的时间。

```
import os
import xlrd
import xlwt
import torch
import datetime
import numpy as np
import torch.nn as nn
import torch.utils.data as Data
import matplotlib.pyplot as plt

torch.manual_seed(1)

PATH ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\总数据.xlsx"
BATCH_SIZE =8
EPOCH =200

class RNN(nn.Module):
    def __init__(self):
        super(RNN, self).__init__()
        self.rnn =nn.LSTM(
            input_size=input_size,
            hidden_size=64,
            num_layers=2,
            batch_first=True,
        )
        self.out =nn.Linear(64, 2)

    def forward(self, x):
        r_out, (h_n, h_c) =self.rnn(x, None)

        out =self.out(r_out[:, -1, :])
        return out

if __name__ =="__main__":
    if not os.path.isdir("./models"):
        os.mkdir("models")

    # load data
    data =xlrd.open_workbook(PATH)
```

```
    table =data.sheet_by_name("因子6")
    input_number =table.nrows -1
    input_size =table.ncols -2

    train_size =input_number
    train_x =torch.zeros((train_size, input_size))
    train_y =torch.zeros((train_size,))

    # transform from excel file to torch tensor
    train_p =0
    for i in range(1, table.nrows):
        for j in range(input_size):
            train_x[train_p, j] =table.cell(i, j +2).value
        train_y[train_p] =table.cell(i, 1).value
        train_p +=1

    # normalize
    '''
    for j in range(input_size):
        mean = train_x[:, j].mean()
        std = train_x[:, j].std()
        train_x[:, j] = (train_x[:, j] - mean) / std
        # train_x[:, j] = train_x[:, j] / mean

        mean = test_x[:, j].mean()
        std = test_x[:, j].std()
        test_x[:, j] = (test_x[:, j] - mean) / std
        # test_x[:, j] = test_x[:, j] / mean
    '''

    # create data loader
    dataset =Data.TensorDataset(train_x, train_y)
    data_loader =Data.DataLoader(
        dataset=dataset,
        batch_size=BATCH_SIZE,
        shuffle=True,
    )

    device =torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model =nn.Sequential(
        nn.Linear(input_size, 64),
        nn.ReLU(),
        nn.Linear(64, 2),
        nn.Sigmoid(),
    ).to(device)
    # model = RNN().to(device)

    optimizer =torch.optim.Adam(model.parameters(), lr=1e-3)
    loss_func =torch.nn.CrossEntropyLoss()

    for epoch in range(EPOCH):
```

```
        loss_all =[]
        for step, (batch_x, batch_y) in enumerate(data_loader):
            x =batch_x.type(torch.FloatTensor).to(device)
            y =batch_y.type(torch.LongTensor).to(device)
            # pred = model.forward(x.view(-1, 1, input_size)) # (batch, time_step, input_size)
            pred =model.forward(x)
            loss =loss_func(pred, y)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            loss_all.append(loss.cpu().item())

        if epoch % 10 ==0:
            # calculate accuracy
            # pred = model.forward(train_x.view(-1, 1, input_size).to(device))
            pred =model.forward(train_x.to(device))
            pred_y =torch.max(pred, 1)[1].cpu().data.numpy()
            target_y =train_y.data.numpy()
            train_acc =float((pred_y ==target_y).astype(int).sum()) /float(target_y.size)
            print("|| Epoch {} | Loss {:.4f} | train_acc {:.4f} ||".format(epoch, np.mean(loss_all),
                                                                train_acc))

    # save model
    model_name ='model_{}.h5'.format(datetime.datetime.now().strftime('%m%d_%H_%M'))
    ckpt_path =os.path.join("./models", model_name)
    ckpt_dict =model.state_dict()
    torch.save(ckpt_dict, ckpt_path)
```

A.2.2 预测代码

二分类神经网络预测代码： Classification_predict.py

输入：格式同 Classification_train.py，内容可以是验证集的数据或附件二经过处理后的数据。

输出：pred[i] 表示第 i 家企业的守约概率。

```
import os
import xlrd
import xlwt
import torch
import datetime
import numpy as np
import torch.nn as nn
import torch.utils.data as Data
import matplotlib.pyplot as plt

torch.manual_seed(1)

PATH ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\问题二\\问题三\\问题二总数据1.xlsx"
MODEL ="./models/model_0913_07_14.h5"
BATCH_SIZE =8
```

```
EPOCH =500

if __name__ =="__main__":
    # load data
    data =xlrd.open_workbook(PATH)
    table =data.sheet_by_name("因子6")
    input_number =table.nrows -1
    input_size =table.ncols -2

    train_size =input_number
    train_x =torch.zeros((train_size, input_size))
    train_y =torch.zeros((train_size,))

    # transform from excel file to torch tensor
    train_p =0
    for i in range(1, table.nrows):
        for j in range(input_size):
            train_x[train_p, j] =table.cell(i, j +2).value
        # train_y[train_p] = table.cell(i, 1).value
        train_p +=1

    # normalize
    '''
    for j in range(input_size):
        mean = train_x[:, j].mean()
        std = train_x[:, j].std()
        train_x[:, j] = (train_x[:, j] - mean) / std
        # train_x[:, j] = train_x[:, j] / mean

        mean = test_x[:, j].mean()
        std = test_x[:, j].std()
        test_x[:, j] = (test_x[:, j] - mean) / std
        # test_x[:, j] = test_x[:, j] / mean
    '''

    # create data loader
    dataset =Data.TensorDataset(train_x, train_y)
    data_loader =Data.DataLoader(
        dataset=dataset,
        batch_size=BATCH_SIZE,
        shuffle=True,
    )

    model =nn.Sequential(
        nn.Linear(input_size, 64),
        nn.ReLU(),
        nn.Linear(64, 2),
        nn.Sigmoid(),
    )

    optimizer =torch.optim.Adam(model.parameters(), lr=1e-3)
```

```
    loss_func =torch.nn.CrossEntropyLoss()

    model.load_state_dict(torch.load(MODEL))
    # pred = model.forward(train_x.view(-1, 1, input_size))
    pred =model.forward(train_x)

    wb =xlwt.Workbook()
    sheet1 =wb.add_sheet("Sheet1")
    for i in range(train_size):
        sheet1.write(i, 0, float(pred[i, 1].data))
    wb.save('temp.xls')
```

A.3 四分类神经网络代码

1. 四分类神经网络验证代码： Classification_validation.py

输入：与支撑材料中总数据.xlsx 形式相同的文件，最右边一列为企业的信誉评级。

输出：以.h5 存储的模型参数和训练集测试集精度随训练代数的变化曲线图。

```
import os
import xlrd
import xlwt
import torch
import datetime
import numpy as np
import torch.nn as nn
import torch.utils.data as Data
import matplotlib.pyplot as plt

torch.manual_seed(1)

PATH ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\总数据.xlsx"
BATCH_SIZE =16
EPOCH =30000

class RNN(nn.Module):
    def __init__(self):
        super(RNN, self).__init__()
        self.rnn =nn.LSTM(
            input_size=input_size,
            hidden_size=64,
            num_layers=2,
            batch_first=True,
        )
        self.out =nn.Linear(64, 4)

    def forward(self, x):
        r_out, (h_n, h_c) =self.rnn(x, None)

        out =self.out(r_out[:, -1, :])
        return out
```

```
33
34 if __name__ =="__main__":
35     if not os.path.isdir("./models"):
36         os.mkdir("models")
37
38     # load data
39     data =xlrd.open_workbook(PATH)
40     table =data.sheet_by_name("四分类")
41     input_number =table.nrows -1
42     input_size =table.ncols -2
43
44     train_size =input_number -input_number //5
45     train_x =torch.zeros((train_size, input_size))
46     train_y =torch.zeros((train_size,))
47     test_x =torch.zeros((input_number -train_size, input_size))
48     test_y =torch.zeros((input_number -train_size,))
49
50     # transform from excel file to torch tensor
51     train_p =0
52     test_p =0
53     for i in range(1, table.nrows):
54         if i % 5 !=0:
55             for j in range(input_size):
56                 train_x[train_p, j] =table.cell(i, j +2).value
57             train_y[train_p] =table.cell(i, 1).value
58             train_p +=1
59         else:
60             for j in range(input_size):
61                 test_x[test_p, j] =table.cell(i, j +2).value
62             test_y[test_p] =table.cell(i, 1).value
63             test_p +=1
64
65     # normalize
66     '''
67     for j in range(input_size):
68         mean = train_x[:, j].mean()
69         std = train_x[:, j].std()
70         train_x[:, j] = (train_x[:, j] - mean) / std
71         # train_x[:, j] = train_x[:, j] / mean
72
73         mean = test_x[:, j].mean()
74         std = test_x[:, j].std()
75         test_x[:, j] = (test_x[:, j] - mean) / std
76         # test_x[:, j] = test_x[:, j] / mean
77     '''
78
79     wb =xlwt.Workbook()
80     sheet1 =wb.add_sheet("train")
81     sheet2 =wb.add_sheet("test")
82     for i in range(train_size):
83         for j in range(input_size):
```

```
            sheet1.write(i, j+1, train_x[i, j].item())
        sheet1.write(i, 0, train_y[i].item())
    for i in range(input_number -train_size):
        for j in range(input_size):
            sheet2.write(i, j+1, test_x[i, j].item())
        sheet2.write(i, 0, test_y[i].item())
    wb.save('train and test_4cla.xls')


    # create data loader
    dataset =Data.TensorDataset(train_x, train_y)
    data_loader =Data.DataLoader(
        dataset=dataset,
        batch_size=BATCH_SIZE,
        shuffle=True,
    )

    device =torch.device("cuda" if torch.cuda.is_available() else "cpu")

    model =nn.Sequential(
        nn.Linear(input_size, 128),
        nn.ReLU(),
        nn.Linear(128, 4),
        nn.Softmax(),
    ).to(device)

    # model = RNN().to(device)

    optimizer =torch.optim.Adam(model.parameters(), lr=1e-3)
    loss_func =torch.nn.CrossEntropyLoss()

    train_ACC =[]
    test_ACC =[]
    in_X =[]

    for epoch in range(EPOCH):
        loss_all =[]
        for step, (batch_x, batch_y) in enumerate(data_loader):
            x =batch_x.type(torch.FloatTensor).to(device)
            y =batch_y.type(torch.LongTensor).to(device)
            # pred = model.forward(x.view(-1, 1, input_size)) # (batch, time_step, input_size)
            pred =model.forward(x)
            loss =loss_func(pred, y)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            loss_all.append(loss.cpu().item())

        if epoch % 10 ==0:
```

```
            # calculate accuracy on testing data
            # pred = model.forward(train_x.view(-1, 1, input_size).to(device))
            pred =model.forward(train_x.to(device))
            pred_y =torch.max(pred, 1)[1].cpu().data.numpy()
            target_y =train_y.data.numpy()
            train_acc =float((pred_y ==target_y).astype(int).sum()) /float(target_y.size)

            # pred = model.forward(test_x.view(-1, 1, input_size).to(device))
            pred =model.forward(test_x.to(device))
            pred_y =torch.max(pred, 1)[1].cpu().data.numpy()
            target_y =test_y.data.numpy()
            accuracy =float((pred_y ==target_y).astype(int).sum()) /float(target_y.size)

            print("|| Epoch {} | Loss {:.4f} | train_acc {:.4f} | test_acc {:.4f} As {} Bs {} Cs {} Ds {
                                                                    } ||".format(
                epoch, np.mean(loss_all), train_acc, accuracy,
                sum((pred_y ==3)*1), sum((pred_y ==2)*1), sum((pred_y ==1)*1), sum(pred_y ==0)*1))

            in_X.append(epoch)
            train_ACC.append(train_acc)
            test_ACC.append(accuracy)

    plt.plot(in_X, train_ACC, color='blue', label='Training Accuracy', linewidth=0.5)
    plt.plot(in_X, test_ACC, color='red', label='Testing Accuracy', linewidth=0.5)
    plt.legend()
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.show()

    model_name ='model_TEST_{}.h5'.format(datetime.datetime.now().strftime('%m%d_%H_%M'))
    ckpt_path =os.path.join("./models", model_name)
    ckpt_dict =model.state_dict()
    torch.save(ckpt_dict, ckpt_path)
```

2. 依照式(35)预测客户流失率代码： pred_loss_rate.py

输入：附件 3 的客户流失率表，形式同总数据.xlsx 中名为四分类的三因子数据表，以.h5 格式存储的模型参数。

输出：pred_loss.xls，其中第 i 行第 j 列的元素表示第 i 家企业在利率等级为 j 时，预测的客户流失率。

```
import xlrd
import xlwt
import torch
import numpy as np
import torch.nn as nn

RATE ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\C\\附件3：银行贷款年利率与客户流失率关系的统计数据.
                                                    xlsx"
DATA ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\问题二\\问题三\\问题二总数据1税0.9.xlsx"
MODEL ="./models/model_0912_20_28.h5"

```

```
rate =xlrd.open_workbook(RATE)
table_rate =rate.sheet_by_name("Sheet1")
N_rate =table_rate.nrows -2
# 第 2 行到第 N_rate + 1 行为有效数据

data =xlrd.open_workbook(DATA)
table_data =data.sheet_by_name("四分类")
N_company =table_data.nrows -1
# 第 1 行到第 N_company 行为有效数据
test_x =torch.zeros((N_company, 3))
for i in range(N_company):
    for j in range(3):
        test_x[i, j] =table_data.cell_value(i +1, j +1)

model =nn.Sequential(
        nn.Linear(3, 128),
        nn.ReLU(),
        nn.Linear(128, 4),
        nn.Softmax(),
    )
pred =model.forward(test_x)
# pred[i, 0] 第 i 家企业属于D级企业的概率
# pred[i, 1] 第 i 家企业属于C级企业的概率
# pred[i, 2] 第 i 家企业属于B级企业的概率
# pred[i, 3] 第 i 家企业属于A级企业的概率

pred_rate =np.zeros((N_rate, N_company))
# pred_rate[i, j] 表示第 i 个利率下, 第 j 家企业的预测客户流失率

i_rate =np.zeros((N_rate,))
loss_rate =np.zeros((N_rate, 4))
for i in range(N_rate):
    i_rate[i] =table_rate.cell(i +2, 0).value
    for j in range(3):
        loss_rate[i, j] =table_rate.cell(i +2, j +1).value
loss_rate[:, 3] =0.
# loss_rate[i, 0] 表示第 i 个利率下, A级的客户流失率
# loss_rate[i, 1] 表示第 i 个利率下, B级的客户流失率
# loss_rate[i, 2] 表示第 i 个利率下, C级的客户流失率
# loss_rate[i, 3] 表示第 i 个利率下, D级的客户流失率

for i in range(N_rate):
    for j in range(N_company):
        pred_rate[i, j] =pred[j, 0] *loss_rate[i, 3] +pred[j, 1] *loss_rate[i, 2] +\
                        pred[j, 2] *loss_rate[i, 1] +pred[j, 3] *loss_rate[i, 0]


wb =xlwt.Workbook()
sheet1 =wb.add_sheet("Sheet1")
for i in range(N_rate):
    for j in range(N_company):
```

```
        sheet1.write(j, i, float(pred_rate[i, j]))
wb.save('pred_loss.xls')
```

3. 验证式(35)合理性代码： Loss_rate.py

输入：附件 3 的客户流失率表，pred_loss_rate.py 的输出表格 pred_loss.xls。

输出：真实客户流失率与预测客户流失率的均方误差。

```
import xlrd
import xlwt
import torch
import numpy as np
import torch.nn as nn

RATE ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\C\\附件3：银行贷款年利率与客户流失率关系的统计数据.xlsx"
DATA ="D:\\MathematicalModeling\\2020国赛\\2020参赛\\问题二\\问题三\\问题二总数据1税0.9.xlsx"
MODEL ="./models/model_0912_20_28.h5"

rate =xlrd.open_workbook(RATE)
table_rate =rate.sheet_by_name("Sheet1")
N_rate =table_rate.nrows -2
# 第 2 行到第 N_rate + 1 行为有效数据

data =xlrd.open_workbook(DATA)
table_data =data.sheet_by_name("四分类")
N_company =table_data.nrows -1
# 第 1 行到第 N_company 行为有效数据
test_x =torch.zeros((N_company, 3))
for i in range(N_company):
    for j in range(3):
        test_x[i, j] =table_data.cell_value(i +1, j +1)

model =nn.Sequential(
        nn.Linear(3, 128),
        nn.ReLU(),
        nn.Linear(128, 4),
        nn.Softmax(),
    )
pred =model.forward(test_x)
# pred[i, 0] 第 i 家企业属于D级企业的概率
# pred[i, 1] 第 i 家企业属于C级企业的概率
# pred[i, 2] 第 i 家企业属于B级企业的概率
# pred[i, 3] 第 i 家企业属于A级企业的概率

pred_rate =np.zeros((N_rate, N_company))
# pred_rate[i, j] 表示第 i 个利率下，第 j 家企业的预测客户流失率

i_rate =np.zeros((N_rate,))
loss_rate =np.zeros((N_rate, 4))
for i in range(N_rate):
    i_rate[i] =table_rate.cell(i +2, 0).value
```

```
    for j in range(3):
        loss_rate[i, j] =table_rate.cell(i +2, j +1).value
loss_rate[:, 3] =0.
# loss_rate[i, 0] 表示第 i 个利率下, A级的流失率
# loss_rate[i, 1] 表示第 i 个利率下, B级的流失率
# loss_rate[i, 2] 表示第 i 个利率下, C级的流失率
# loss_rate[i, 3] 表示第 i 个利率下, D级的流失率

for i in range(N_rate):
    for j in range(N_company):
        pred_rate[i, j] =pred[j, 0] *loss_rate[i, 3] +pred[j, 1] *loss_rate[i, 2] +\
                        pred[j, 2] *loss_rate[i, 1] +pred[j, 3] *loss_rate[i, 0]


wb =xlwt.Workbook()
sheet1 =wb.add_sheet("Sheet1")
for i in range(N_rate):
    for j in range(N_company):
        sheet1.write(j, i, float(pred_rate[i, j]))
wb.save('pred_loss.xls')
```

A.4 信贷决策求解代码

solve_P2.py

输入：pred_loss_rate.py 的输出表格 pred_loss.xls。

输出：银行信贷决策时优先级排在前 100 的企业的优先级顺序。

```
import xlrd
import xlwt
import numpy as np
from scipy import optimize as op
import matplotlib.pyplot as plt

loss =xlrd.open_workbook("pred_loss.xls").sheet_by_name("Sheet1")
N_company =loss.nrows -1
N_rates =loss.ncols -1
rates =np.zeros((N_rates,))
for i in range(N_rates):
    rates[i] =float(loss.cell(0, i +1).value)

total_data =xlrd.open_workbook("D:\\MathematicalModeling\\2020国赛\\2020参赛\\问题二\\问题三\\问题二总
                                数据1税0.9.xlsx").sheet_by_name("Sheet1")
pred_Y =np.zeros((N_company,))
for i in range(N_company):
    pred_Y[i] =float(total_data.cell(i +1, 12).value)

loss_rate =np.zeros((N_company, N_rates))
for i in range(N_company):
    for j in range(N_rates):
        loss_rate[i, j] =float(loss.cell(i +1, j +1).value)
```

```
23
24  # solve r[i]
25  # r[i] 表示给第 i 家公司贷款的比例
26  max_r =np.zeros((N_company,))
27  last_opt =0
28  A_ub =np.zeros((1, N_company))
29  B_ub =np.array([0])
30
31  wb =xlwt.Workbook()
32  sheet1 =wb.add_sheet("Sheet1")
33  for _ in range(100):
34      objective =0
35      for rate in range(N_rates):
36          # maximize \sum_i {r[i] * (1 - loss_rate[i, rate]) * (pred_Y[i] * (1+rates[rate]) - (1 - pred_Y
                                                                    [i]))} - 1
37          # subject to \sum_i {r[i]} = 1
38          c =np.zeros((N_company,))
39          if _ >0:
40              A_ub[0, last_opt] =1
41          for i in range(N_company):
42              c[i] =(1 -loss_rate[i, rate]) *(pred_Y[i] *(1 +rates[rate]) -(1 -pred_Y[i]))
43          A_eq =np.ones((1, N_company))
44          B_eq =np.array([1])
45          now =op.linprog(-c, A_ub=A_ub, b_ub=B_ub, A_eq=A_eq, b_eq=B_eq, method='simplex')
46          if -now['fun'] >objective:
47              objective =-now['fun']
48              max_r =now['x']
49              ans_rate =rates[rate]
50      # print(max_r)
51      for i in range(N_company):
52          if max_r[i] ==1.:
53              last_opt =i
54              print(total_data.cell(i +1, 0).value)
55              sheet1.write(_ //10, _ % 10, total_data.cell(i +1, 0).value)
56  wb.save("temp4.xls")
```

B 因子分析表及问题二、问题三求得的银行最优解序列

表 12: 问题一评价企业各项指标描述性统计

指标类别	指标名称	平均值	标准偏差	样本数
现金流指标	净现金流增长率	-0.0558	10.2795	121
	现金流波动性	2.2541	2.0055	121
	现金流入流出比	161.86	842.64	121
企业规模指标	销售总额	8.83E+06	3.75E+07	121
	净税	3.43E+06	3.64E+07	121
销售质量指标	利润率	148.67	826.97	121
	退货比例	0.5848	2.8867	121
供求稳定性指标	上游供应稳定性指标 1	0.6080	0.3228	121
	上游供应稳定性指标 2	0.3472	0.3409	121
	下游需求稳定性指标 1	0.6300	0.3103	121
	下游需求稳定性指标 2	0.3772	0.3334	121

表 13: 问题一因子分析旋转后成分矩阵

指标	成分					
	1	2	3	4	5	6
净现金流增长率	0.005	0.064	0.031	-0.010	-0.026	0.991
现金流波动性	-0.103	0.534	0.387	-0.027	-0.451	-0.107
现金流入流出比	0.990	-0.014	-0.046	-0.004	-0.006	0.006
销售总额	-0.024	0.088	0.261	0.831	0.081	-0.024
净税	-0.015	-0.034	0.164	-0.867	0.115	-0.013
利润率	0.991	-0.036	-0.007	-0.004	-0.004	0.002
退货比例	-0.032	0.223	-0.025	-0.045	0.895	-0.049
上游供应稳定性指标 1	0.047	0.148	0.912	0.018	-0.039	0.069
上游供应稳定性指标 2	-0.095	0.179	0.904	0.054	-0.036	-0.024
下游需求稳定性指标 1	0.092	0.871	0.237	0.017	0.131	0.045
下游需求稳定性指标 2	-0.112	0.895	0.060	0.134	0.157	0.074

[1] 提取方法：主成分分析法。

[2] 旋转方法：凯撒正态化最大方差法。

[3] 旋转在 6 次迭代后收敛。

表 14: 问题一因子分析总方差解释表

因子	初始特征值			提取载荷平方和			旋转载荷平方和		
	总计	方差占比%	累积%	总计	方差占比%	累积%	总计	方差占比%	累积%
1	2.839	25.805	25.805	2.839	25.805	25.805	2.006	18.239	18.239
2	1.963	17.850	43.655	1.963	17.850	43.655	1.962	17.839	36.078
3	1.434	13.037	56.692	1.434	13.037	56.692	1.957	17.791	53.868
4	1.397	12.700	69.392	1.397	12.700	69.392	1.467	13.333	67.201
5	1.008	9.163	78.555	1.008	9.163	78.555	1.069	9.719	76.921
6	0.829	7.536	86.091	0.829	7.536	86.091	1.009	9.170	86.091
7	0.604	5.494	91.585						
8	0.464	4.219	95.803						
9	0.230	2.092	97.896						
10	0.203	1.841	99.737						
11	0.029	0.263	100.000						

[1] 提取方法：主成分分析法。

表 15: 问题一 Logistic 回归测试集结果

实际 Y_i	因子 1	因子 2	因子 3	因子 4	因子 5	因子 6	预测值	预测结果
0	-0.14107	0.02868	1.2565	-0.02159	0.57592	0.14376	0.933763	FALSE
0	-0.23982	-1.24076	-0.10075	0.03274	0.10508	4.52816	0.313209	TRUE
0	-0.20084	2.3994	-0.53863	-0.20093	9.10842	-0.41495	0.769416	FALSE
0	-0.20984	-1.09794	-1.32417	0.0472	-0.19704	-0.09088	0.450464	TRUE
0	-0.29874	-1.51504	-0.87402	0.07474	-0.07305	0.28282	0.408811	TRUE
1	-0.06942	1.7199	-0.73238	0.01728	-0.31428	0.01859	0.931372	TRUE
1	-0.15861	0.07289	0.78346	0.14343	0.17118	0.07288	0.916467	TRUE
1	-0.124	0.7894	0.63384	-0.06314	-0.449	-0.06906	0.953544	TRUE
0	1.45475	0.78229	-1.54707	0.0258	-0.08389	0.21444	0.924845	FALSE
1	-0.18517	-0.05655	0.26153	-0.08475	-0.2458	-0.17161	0.878706	TRUE
1	-0.19875	0.72925	-1.33949	0.0295	-0.14376	0.04845	0.762192	TRUE
1	2.76519	-0.93405	0.00642	0.00887	-0.00691	0.13237	0.975827	TRUE
1	-0.13641	0.29137	0.89816	-0.11668	0.00258	0.1492	0.935375	TRUE
0	-0.18326	1.27135	-1.24965	-0.11657	-0.54828	0.39277	0.83859	FALSE
1	-0.08357	-0.70528	0.21253	-0.08444	-0.09744	-0.06107	0.812005	TRUE
1	-0.14003	0.95029	0.34754	-0.14775	-0.28034	-0.06091	0.9451	TRUE
1	-0.07465	-0.6914	1.61509	-0.06726	0.4018	-0.04373	0.926756	TRUE
1	-0.16387	0.00288	0.36197	-0.04595	0.11504	0.00132	0.880948	TRUE
1	-0.26724	-1.12689	0.59381	-0.05842	-0.4472	-0.09866	0.797851	TRUE
1	-0.19145	-0.31136	1.24149	0.11584	0.01693	-0.06454	0.925599	TRUE
1	-0.21623	0.02572	-0.94412	-0.03241	-0.09186	0.00664	0.716357	TRUE
1	-0.17438	0.34817	-1.6172	-0.01352	-0.14857	0.00195	0.661645	TRUE
1	-0.13888	-0.12439	0.18746	-0.08399	0.03058	0.01892	0.856937	TRUE
1	-0.24705	-0.01461	-1.51704	0.00121	-0.00924	0.14691	0.581956	TRUE

表 16: 问题二评价企业各项指标描述性统计

指标类别	指标名称	平均值	标准偏差	分析个案数
现金流指标	净现金流增长率	-1.2757	25.4835	302
	现金流波动性	0.7871	0.6783	302
	现金流入流出比	20.0533	111.3937	302
企业规模指标	销售总额	50766821.84	95109752.30	302
	净税	1555216.64	4088761.99	302
销售质量指标	利润率	127.8567	673.0748	302
	退货比例	0.0214	0.0375	302
供求稳定性指标	上游供应稳定性指标 1	0.5851	0.3002	302
	上游供应稳定性指标 2	0.3059	0.2977	302
	下游需求稳定性指标 1	0.3959	0.3350	302
	下游需求稳定性指标 2	0.1627	0.2657	302

表 17: 问题二 KMO 和巴特利特球形度检验

KMO 取样适切性量数		0.718
巴特利特球形度检验	近似卡方	1009.219
	自由度	55
	显著性	0.000

表 18: 问题二因子分析总方差解释表

因子	初始特征值			提取载荷平方和			旋转载荷平方和		
	总计	方差占比%	累积%	总计	方差占比%	累积%	总计	方差占比%	累积%
1	2.176	19.783	19.783	2.176	19.783	19.783	1.893	17.212	17.212
2	1.901	17.285	37.068	1.901	17.285	37.068	1.805	16.412	33.624
3	1.608	14.619	51.687	1.608	14.619	51.687	1.674	15.220	48.844
4	1.300	11.821	63.508	1.300	11.821	63.508	1.600	14.546	63.390
5	1.019	9.268	72.776	1.019	9.268	72.776	1.008	9.160	72.551
6	0.978	8.888	81.664	0.978	8.888	81.664	1.002	9.113	81.664
7	0.859	7.807	89.470						
8	0.408	3.710	93.180						
9	0.356	3.237	96.417						
10	0.286	2.602	99.018						
11	0.108	0.982	100.000						

[1] 提取方法：主成分分析法。

表 19: 问题二因子分析旋转后成分矩阵

指标	成分					
	1	2	3	4	5	6
净现金流增长率	0.038	0.001	0.033	0.000	0.005	0.990
现金流波动性	0.088	-0.470	0.174	-0.243	-0.065	-0.122
现金流入流出比	0.966	-0.080	-0.007	-0.008	-0.035	0.020
销售总额	-0.018	0.107	0.133	0.861	0.061	0.005
净税	0.010	-0.020	0.090	0.880	-0.052	-0.007
利润率	0.970	-0.025	-0.001	-0.009	0.043	0.022
退货比例	0.007	-0.007	0.029	0.008	0.995	0.006
上游供应稳定性指标 1	0.040	0.915	0.020	-0.011	-0.009	-0.021
上游供应稳定性指标 2	-0.085	0.853	0.167	-0.043	-0.049	-0.068
下游需求稳定性指标 1	0.002	0.041	0.886	0.129	0.004	0.005
下游需求稳定性指标 2	-0.011	0.006	0.896	0.083	0.028	0.028

[1] 提取方法：主成分分析法。

[2] 旋转方法：凯撒正态化最大方差法。

[3] 旋转在 4 次迭代后收敛。

表 20: 附件一四分类因子分析总方差解释表

因子	初始特征值			提取载荷平方和			旋转载荷平方和		
	总计	方差占比%	累积%	总计	方差占比%	累积%	总计	方差占比%	累积%
1	2.147	35.786	35.786	2.147	35.786	35.786	2.048	34.138	34.138
2	1.411	23.511	59.297	1.411	23.511	59.297	1.470	24.502	58.640
3	1.060	17.666	76.963	1.060	17.666	76.963	1.099	18.322	76.963
4	0.625	10.409	87.372						
5	0.529	8.810	96.182						
6	0.229	3.818	100.000						

[1] 提取方法：主成分分析。

表 21: 附件一四分类因子分析旋转后成分矩阵

指标	成分		
	1	2	3
现金流波动性	0.645	-0.017	-0.541
销售总额	0.186	0.828	-0.010
净税	0.057	-0.871	0.033
退货比例	0.239	-0.051	0.863
下游需求稳定性指标 1	0.892	0.029	0.132
下游需求稳定性指标 2	0.862	0.147	0.207

[1] 提取方法：主成分分析法。

[2] 旋转方法：凯撒正态化最大方差法。

[3] 旋转在 5 次迭代后收敛。

表 22: 附件二四分类 KMO 和巴特利特球形度检验

KMO 取样适切性量数		0.748
巴特利特球形度检验	近似卡方	308.109
	自由度	15
	显著性	0.000

表 23: 附件二四分类因子分析总方差解释表

成分	初始特征值			提取载荷平方和			旋转载荷平方和		
	总计	方差占比%	累积%	总计	方差占比%	累积%	总计	方差占比%	累积%
1	1.978	32.972	32.972	1.978	32.972	32.972	1.702	28.371	28.371
2	1.307	21.779	54.751	1.307	21.779	54.751	1.580	26.333	54.704
3	1.000	16.672	71.422	1.000	16.672	71.422	1.003	16.718	71.422
4	0.952	15.863	87.285						
5	0.412	6.865	94.150						
6	0.351	5.850	100.000						

[1] 提取方法：主成分分析。

表 24: 附件二四分类因子分析旋转后成分矩阵

指标	成分		
	1	2	3
现金流波动性	0.158	-0.384	-0.039
销售总额	0.197	0.849	0.048
净税	0.161	0.840	-0.068
退货比例	0.025	0.023	0.997
下游需求稳定性指标 1	0.895	0.073	0.003
下游需求稳定性指标 2	0.900	0.023	0.028

[1] 提取方法：主成分分析法。

[2] 旋转方法：凯撒正态化最大方差法。

[3] 旋转在 3 次迭代后收敛。

表 25: 突发事件强度参数 $t = 0.5$ 时，银行的最优解序列

E128	E216	E278	E332	E196	E152	E185	E297	E148	E240
E139	E193	E243	E326	E246	E127	E254	E182	E155	E175
E167	E402	E315	E292	E324	E379	E342	E180	E321	E262
E227	E194	E241	E186	E270	E263	E242	E142	E304	E260
E359	E249	E195	E312	E289	E314	E284	E178	E333	E269
E295	E141	E211	E334	E164	E341	E313	E290	E163	E146
E365	E369	E287	E189	E282	E149	E136	E348	E154	E298
E306	E378	E138	E220	E171	E172	E288	E187	E305	E153
E329	E208	E293	E392	E299	E210	E145	E140	E181	E168
E256	E174	E201	E357	E281	E285	E235	E373	E387	E401

表 26: 突发事件强度参数 $t = 0.7$ 时，银行的最优解序列

E231	E357	E343	E365	E244	E410	E406	E316	E321	E155
E352	E295	E366	E227	E361	E395	E388	E409	E186	E348
E408	E153	E264	E413	E384	E237	E299	E373	E401	E346
E420	E398	E266	E330	E421	E412	E309	E341	E241	E417
E334	E403	E282	E422	E387	E222	E333	E273	E327	E424
E390	E335	E313	E229	E375	E414	E360	E349	E419	E394
E338	E364	E399	E369	E285	E350	E281	E345	E252	E290
E354	E325	E247	E329	E280	E283	E288	E256	E298	E392
E306	E314	E301	E423	E207	E258	E319	E291	E347	E359
E287	E294	E183	E293	E277	E336	E402	E312	E378	E188

表 27: 突发事件强度参数 $t = 0.8$ 时，银行的最优解序列

E357	E231	E185	E165	E332	E321	E240	E152	E297	E139
E148	E227	E155	E127	E295	E182	E326	E196	E167	E254
E128	E246	E263	E278	E292	E243	E237	E180	E189	E262
E175	E315	E304	E208	E242	E216	E193	E324	E270	E343
E178	E244	E249	E379	E211	E138	E136	E195	E342	E164
E284	E289	E141	E260	E269	E314	E410	E133	E229	E264
E168	E194	E146	E183	E333	E401	E312	E287	E142	E316
E154	E313	E359	E145	E369	E220	E290	E298	E406	E187
E222	E334	E392	E256	E341	E402	E327	E305	E361	E241
E293	E299	E282	E306	E188	E210	E352	E288	E235	E309

表 28: 突发事件强度参数 $t = 0.9$ 时，银行的最优解序列

E128	E196	E194	E146	E142	E175	E136	E149	E152	E168
E158	E163	E173	E131	E181	E255	E171	E242	E141	E202
E233	E210	E211	E127	E189	E217	E154	E293	E187	E172
E304	E261	E318	E178	E301	E253	E220	E283	E314	E354
E201	E325	E195	E279	E174	E280	E345	E225	E164	E419
E334	E329	E177	E290	E306	E364	E145	E291	E188	E294
E319	E277	E180	E235	E303	E423	E347	E336	E252	E335
E288	E341	E360	E402	E247	E292	E140	E207	E183	E258
E284	E387	E282	E333	E403	E289	E269	E167	E182	E359
E273	E203	E399	E313	E417	E264	E302	E307	E390	E241

表 29: 突发事件强度参数 $t = 1.1$ 时，银行的最优解序列

E128	E196	E194	E142	E146	E136	E131	E152	E175	E149
E168	E158	E173	E163	E127	E255	E233	E189	E133	E139
E181	E217	E150	E171	E164	E177	E202	E207	E183	E354
E195	E261	E237	E145	E155	E419	E345	E174	E141	E210
E225	E229	E252	E364	E235	E148	E247	E242	E319	E264
E188	E208	E335	E203	E307	E423	E318	E178	E360	E211
E172	E140	E258	E201	E403	E220	E303	E294	E417	E291
E309	E182	E279	E273	E277	E154	E399	E421	E187	E336
E349	E302	E330	E338	E301	E390	E420	E375	E347	E398
E280	E412	E346	E167	E384	E408	E325	E350	E422	E409

表 30: 突发事件强度参数 $t = 1.2$ 时，银行的最优解序列

E146	E149	E181	E148	E163	E185	E136	E240	E167	E182
E173	E158	E293	E168	E194	E202	E187	E297	E142	E304
E180	E262	E154	E326	E210	E243	E255	E178	E242	E211
E301	E139	E292	E196	E283	E325	E246	E280	E172	E253
E314	E284	E289	E233	E195	E329	E318	E332	E141	E217
E254	E269	E402	E171	E261	E288	E291	E387	E220	E306
E290	E279	E354	E324	E294	E315	E345	E303	E419	E225
E277	E364	E399	E249	E335	E360	E174	E390	E334	E342
E336	E319	E394	E347	E403	E423	E260	E175	E252	E417
E333	E341	E421	E313	E177	E287	E281	E247	E164	E273

表 31: 突发事件强度参数 $t = 1.3$ 时，银行的最优解序列

E406	E395	E244	E316	E366	E352	E410	E270	E222	E343
E264	E309	E237	E229	E361	E164	E208	E128	E327	E295
E249	E321	E183	E155	E379	E256	E145	E357	E414	E258
E227	E235	E194	E207	E220	E188	E195	E174	E231	E175
E196	E163	E392	E266	E189	E330	E285	E149	E142	E338
E172	E177	E210	E347	E298	E336	E315	E385	E178	E247
E279	E305	E146	E187	E181	E154	E225	E277	E259	E211
E193	E369	E299	E281	E139	E423	E287	E312	E202	E252
E273	E278	E173	E203	E168	E261	E294	E318	E158	E255
E269	E136	E233	E280	E253	E262	E148	E140	E283	E288

表 32: 突发事件强度参数 $t = 1.4$ 时，银行的最优解序列

E357	E231	E321	E295	E227	E401	E380	E339	E343	E410
E244	E379	E385	E406	E237	E316	E250	E155	E264	E361
E395	E366	E352	E309	E270	E216	E414	E229	E338	E222
E330	E365	E266	E207	E327	E388	E183	E398	E409	E420
E408	E384	E346	E413	E412	E186	E273	E247	E422	E421
E252	E424	E417	E375	E403	E349	E350	E241	E423	E335
E360	E419	E364	E391	E312	E345	E256	E390	E399	E354
E258	E319	E307	E336	E347	E277	E225	E302	E294	E372
E373	E370	E177	E342	E291	E369	E299	E392	E324	E285
E308	E188	E303	E402	E208	E261	E359	E217	E279	E260

表 33: 突发事件强度参数 $t = 1.5$ 时，银行的最优解序列

E357	E321	E295	E231	E227	E401	E343	E395	E410	E406
E309	E244	E330	E264	E338	E316	E229	E266	E419	E252
E366	E247	E354	E217	E345	E364	E335	E273	E207	E403
E360	E417	E423	E233	E421	E319	E158	E349	E307	E375
E420	E412	E352	E398	E350	E422	E225	E346	E384	E408
E414	E294	E409	E183	E388	E255	E399	E302	E261	E413
E177	E277	E173	E391	E336	E303	E237	E390	E291	E424
E361	E222	E347	E385	E318	E258	E372	E379	E155	E168
E279	E188	E365	E373	E370	E308	E235	E136	E202	E327
E250	E270	E394	E174	E280	E145	E146	E285	E387	E256

表 34: 税额减免率 $t_{tax} = 0.4$ 时，银行的最优解序列

E128	E196	E216	E146	E158	E217	E173	E255	E149	E233
E194	E136	E354	E168	E419	E345	E142	E364	E335	E202
E252	E319	E360	E261	E403	E247	E181	E423	E307	E417
E225	E318	E273	E303	E294	E421	E291	E349	E330	E399
E277	E302	E420	E398	E412	E266	E336	E338	E301	E375
E350	E279	E346	E325	E390	E422	E384	E283	E408	E280
E388	E347	E409	E387	E365	E177	E373	E293	E163	E186
E413	E210	E207	E253	E394	E424	E187	E329	E188	E153
E278	E174	E348	E281	E241	E258	E288	E306	E235	E154
E282	E395	E309	E299	E145	E150	E220	E285	E370	E308

表 35: 税额减免率 $t_{tax} = 0.3$ 时，银行的最优解序列

E128	E196	E152	E139	E231	E357	E148	E127	E155	E194
E146	E136	E142	E321	E175	E227	E168	E295	E182	E254
E158	E240	E167	E185	E149	E163	E131	E237	E173	E189
E242	E178	E304	E264	E141	E195	E278	E211	E262	E379
E410	E343	E233	E255	E229	E183	E270	E164	E180	E207
E181	E243	E249	E138	E174	E145	E177	E297	E244	E284
E171	E309	E235	E315	E210	E172	E154	E220	E208	E188
E269	E222	E326	E202	E187	E258	E246	E406	E133	E289
E314	E140	E203	E217	E395	E316	E292	E327	E256	E193
E225	E366	E290	E293	E201	E247	E261	E361	E318	E352

表 36: 税额减免率 $t_{tax} = 0.2$ 时，银行的最优解序列

E165	E357	E321	E231	E128	E196	E227	E295	E155	E146
E136	E142	E194	E149	E158	E168	E173	E379	E163	E401
E343	E255	E175	E233	E278	E181	E410	E217	E244	E237
E202	E189	E270	E177	E264	E210	E207	E406	E261	E164
E225	E174	E354	E145	E183	E247	E188	E345	E318	E229
E252	E195	E235	E419	E171	E316	E423	E131	E172	E258
E364	E220	E309	E279	E294	E335	E395	E360	E277	E319
E291	E336	E330	E273	E403	E347	E187	E303	E417	E399
E266	E421	E154	E338	E366	E139	E178	E307	E390	E148
E420	E349	E203	E249	E222	E301	E398	E302	E412	E280

表 37: 税额减免率 $t_{tax} = 0.1$ 时，银行的最优解序列

E146	E136	E158	E168	E173	E196	E233	E255	E142	E149
E194	E207	E217	E163	E177	E181	E309	E128	E202	E229
E225	E183	E247	E210	E264	E261	E174	E188	E252	E187
E235	E338	E258	E154	E145	E330	E423	E293	E354	E395
E220	E279	E318	E242	E345	E172	E419	E336	E304	E347
E277	E288	E290	E307	E329	E294	E141	E314	E281	E364
E211	E266	E285	E253	E180	E410	E306	E283	E301	E319
E171	E335	E189	E280	E325	E273	E406	E360	E291	E284
E343	E313	E289	E269	E303	E287	E403	E334	E299	E256
E259	E417	E298	E244	E282	E387	E341	E292	E401	E399