基于多模型量化分析的信贷策略系统

摘要

中小微企业是银行的重要客户资源，但由于大多中小微企业发展不完善等各种不利因素，使其在融资过程中面临各种困难。本文站在银行的角度，目标是使银行的利润最大化，约束条件包括年度信贷总额、对每家企业的贷款额度、贷款利率，贷款期限设定为一年，并且假定政府的相关政策没有变化。利用企业每日的进项、销项数据，挖掘影响中小微企业贷款风险的各种因素，将银行信贷风险分为企业还款意愿和企业还款能力两项指标，其中还款意愿由企业信誉评级量化，企业还款能力包括企业实力，稳定的供求关系，上下游企业影响力。并建立量化指标为构建量化策略提供主要支撑。

针对问题一，首先利用附件 1 中 123 家企业的数据提取出“企业实力”、“上游企业影响力”、“下游企业影响力”、“供求关系稳定性”、“信誉评级”、“是否违约”六个与信贷风险有关的特征，将前 5 个特征输入决策树模型作为自变量，“是否违约”作为因变量，经过交叉验证多轮迭代，反复调节参数，经过验证集的验证，准确率可达到 98%。然后我们将模型中各个特征的权重输出，进而利用这些权重与归一化后的数据做内积经过逻辑回归模型的输入。逻辑回归的输出是一个介于 0 和 1 之间的概率值，将其看作是该企业违约概率，即量化后的信贷风险评分。根据“是否违约”字段调节逻辑回归的阈值为 0.9 时，模型的准确率较高。最后经过线性规划模型，综合前两个模型的量化分析结果，给出满足约束条件时每个企业的贷款额度和贷款利率。

针对问题二，与问题一相比 302 家企业的数据缺少“信誉评级”和“是否违约”字段，而且该银行确定了年度信贷总额度为 1 亿元。最后决定利用准确率最好的随机森林模型预测得到 302 家企业的“信誉评级”。将其带入线性规划模型中，得到 1 亿元的分配策略，即是否发放贷款、贷款金额、贷款利率。

针对问题三，为了提高模型的抗干扰能力，即处理一些突发因素，我们在问题一模型体系的基础上添加了“前期工作”，“前期工作”是指引入监控变量，建立“滑动平均动量监控“模型，实现对风险的自动监控。该模型和问题一模型体系的交互机理是：若检测到异常，则重新收集特征（比如企业类型，行业类型），丰富特征集合，重启问题一模型体系，再次训练，分配量化指标权重；若没有检测到异常，则使用问题一的模型体系得到信贷策略。我们创造性地提出了“滑动平均动量监控模型”，该模型强依赖于时间序列，能够实现信贷策略的自动指定算法，具有较强的鲁棒性。

关键字： 信贷风险量化；决策树；逻辑回归；信贷政策；线性规划；滑动平均监控

一、 问题重述

由于中小微企业自身的诸多不利因素给其融资造成了诸多困难，银行利用中小微企业的日流量数据建立数学模型得出信贷策略。

1. 量化分析 123 家中小微企业的信贷风险并给出银行在年度信贷总额不变时对企业的信贷策略；
2. 量化分析 302 家中小微企业的信贷风险并给出银行在年度信贷总额为 1 亿元时对企业的信贷策略；
3. 考虑 302 家中小微企业的信贷风险和可能出现的突发因素对企业的影响，给出银行在年度信贷总额为 1 亿元时对企业的信贷调整策略。

二、 问题分析

针对问题一中的量化分析，我们首先考虑在实际情况中对银行收益威胁最大的就是贷款人不按时归还应还款项，即贷款人违约，因此我们将企业的违约概率作为信贷风险量化的衡量标准。为降低信贷风险，银行需要考虑借款人的还款意愿以及还款能力，基于此，我们从附件 1 中归纳提取出 6 个相关特征变量——“进项价税合计”，“销项价税合计”，“企业上游影响力”，“企业下游影响力”，” 企业供求稳定性标准差 “，“信誉等级”，“是否违约”。其中“是否违约”就是企业违约概率的一种表现形式。为确定相关特征变量对企业违约概率的影响程度，我们利用决策树模型获得各特征变量的权重，再将其线性组合输入逻辑回归模型中得到企业的信用风险评分，该评分越高，信贷风险就越高。

针对问题一中的信贷策略，我们首先根据客户流失率确定不同信誉评级的贷款年利率，之后利用自然语言处理技术将企业按照不同的信誉评级和企业类型分类，并计算出每一类企业的坏账率，即违约的企业占同类企业的比例，根据企业的违约概率、信誉评级、企业实力、坏账率、不同利率下的客户流失率给出银行的信贷策略构建线性规划模型，信贷策略包括是否贷款、贷款金额以及贷款年利率。

针对问题二的量化分析，因为 302 家企业数据中缺少“信誉评级”这一特征变量，所以我们利用附件 1 数据训练新的决策树模型，预测 302 家企业的“信誉评级”。因为“信誉评级”与问题一的“违约概率”有相似性，我们除了上述的 6 个相关特征变量外还挖掘出新的两个特征变量——“企业三年内交易频次”、“作废发票比例”。基于此，就可以在问题一的基础上进行 302 家企业的信贷风险量化分析。

针对问题二中的信贷策略，利用问题一的线性规划模型，结合问题二实际，在固定

的年度贷款总额为一亿元时，求解目标函数，得出具体每个公司的贷款额度和贷款利率。

针对问题三，每当突发事件发生时，就会产生影响企业信贷风险评分的新的因素，或需要将已有因素重新组合，以新冠肺炎为例，所属行业是一个新的影响因素。我们利用自然语言处理技术提取出每个企业所处的行业作为一个新的特征变量，并将这个变量与“企业类型”一起放入问题一的决策树模型中，重新计算各个特征变量的权重，再结合社会实际情况进行解释说明。

三、 模型假设

- 每家上下游企业的影响力无差异；
- 计算企业利润时不考虑诸如所得税的各项税种，认为“总利润 = 总销项价税合计-总进项价税合计”；
- 企业在开具发票或收入发票的同时完成金钱交付，不存在之后付款的情况；
- 所有企业的生产周期都为一个月。

四、 名词解释与符号说明

4.1 名词解释

坏账率：某一类企业会违约的概率比例，对于同一类企业这是一个固定的值，我们将企业分类后按照附件 1 中“是否违约”显示的违约企业个数除以该类企业总企业数，把其作为该类企业的坏账率。

4.2 符号说明

符号	意义
α_0	进项价税合计
α_1	销项价税合计
β_0	企业上游影响力
β_1	企业下游影响力
σ	企业供求稳定性标准差
γ	信誉评级
M	贷款额度
R	利率
δ	是否违约
P	信贷风险违约概率

W	决策树输出的权重
V	坏账率
F	客户流失率
S	年度信贷总额
A	企业类别
I	企业平均年利润
m	企业月利润移动平均
ε	监控阈值

五、 附件一数据分析及预处理

5.1 数据分析

附件 1 数据包含三张表分别为“企业信息”、“进项发票信息”、“销项发票信息”。“企业信息”包含“企业代号”、“企业名称”、“信誉评级”、“是否违约”四个变量，与银行对某一企业的信用评分有关。“进项发票信息”包含“企业代号”、“开票日期”、“销方单位代号”、“价税合计”、“发票状况”等八个变量，是企业向上游企业购买产品的记录。“销项发票信息”包含与“进项发票信息”对应的八个变量，是企业向下游企业售出产品的记录。

5.2 信贷风险影响特征量化

当银行为中小微企业提供贷款时，需要考虑一个企业的还款意愿与还款能力，一个企业的还款意愿可以用该企业的信誉评级衡量,信誉等级越高，违约概率越低，越容易获得银行贷款同时可以得到更低的利率；一个企业的还款能力需要考虑该企业的企业实力、供求关系、上下游企业影响力，企业实力越高、供求关系越稳定、上下游企业影响力越大说明该企业的还款能力越强，发展状况越稳定，该企业越容易获得银行贷款，且获得的贷款金额较大。这些因素都会涉及银行的信贷风险，具体层次划分如图1所示：

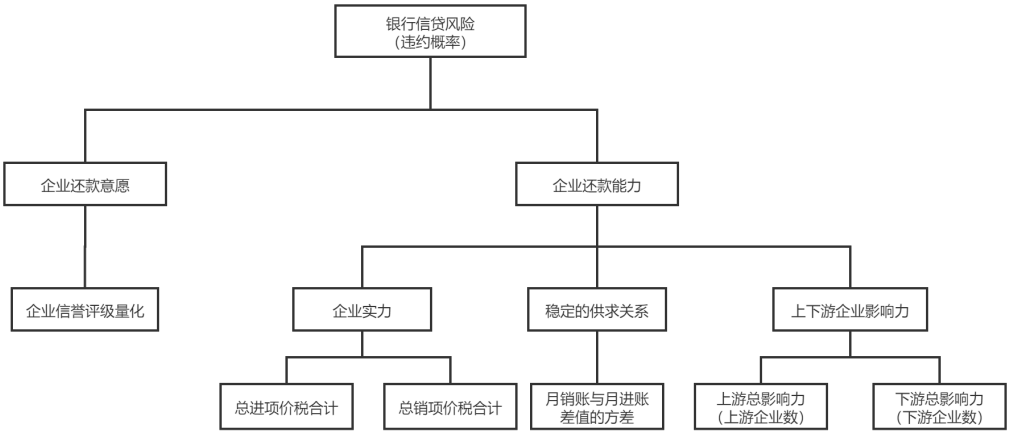

图 **1**　信贷风险影响因素层次图

5.2.1 企业实力量化

为量化企业实力，我们将企业总的进账价税合计与总的销账价税合计作为衡量依据，即将“进项发票信息”和“销项发票信息”两张表中“价税合计”的所有数据按照企业加总。处理后的部分数据如表1所示。

表 **1**　企业实力量化部分结果

企业代号	进项价税合计	销项价税合计
E1	6637942028.2900	4698633440.6000
E10	5417961.6400	351461531.5000
E42	11769.0000	26235453.6099
E61	932031.1800	18687803.7999
E97	19880.0000	1000542.90000

5.2.2 上下游影响力量化

根据我们的假设，每个上下游企业的影响力相同，通过计算上下游企业个数可以判断某一家企业的总体上下游企业影响力，总体上下游企业影响力越大的企业越容易获得银行贷款。因此我们根据“进项发票信息”和“销项发票信息”两张表中的“企业代号”、“销方单位代号”和“购方单位代号”分别统计出每一家企业的上游、下游企业个数，作为两个新特征变量。处理后的部分数据如下表2所示。

表 **2**　上下游企业影响力量化部分结果

	E1	E10	E42	E61	E97
企业上游影响力	436	19	367	486	8
企业下游影响力	352	103	48	13	19

5.2.3 供求关系量化

为量化企业供求关系的稳定情况，我们将企业每月销账总额与进账总额的差值的标准差作为衡量企业供求关系是否稳定的标准，根据我们的假设，这一标准即企业月利润的标准差。我们先将“进项发票信息”和“销项发票信息”两张表按“企业代号”与“开票日期”合并，若企业当天无进账或销账则用 0 填充当天“价税合计”的缺失值。之后将每个企业每日的“销账价税合计”的总额减去当天的“进账价税合计”总额，最后按月合并计算总额、求出每一个企业按月合并总额的标准差。其中 $E1$ 和 $E100$ 两家企业的供求情况如图2所示。

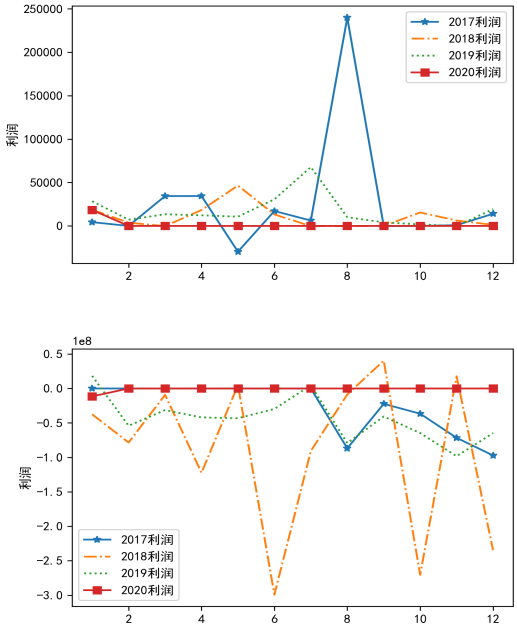

图 **2**　两家企业的供求关系图

图2上为企业 $E100$ 的供求情况，从图中可以看出该企业总体发展比较平稳，其销账与进账的差值仅在 2017 年 8 月突然增加，但 9 月又回归正常水平。整体上该企业保持着稳定的供求关系，证明该企业有稳定的还款能力，银行可以借贷给该企业能力范围内的金额。

图2下为企业 $E1$ 的供应情况，从图中可以看出该企业销账与进账的差值波动较大，

尤其是 2018 年。同时从图中可以看出该企业近 3 年内都是负盈利，说明该企业有可能处于破产边缘，或者有做假账的嫌疑，在这种情况下，银行需要慎重考虑是否借款给该企业。

5.2.4 信誉评级量化

信誉评级是衡量一个企业还款意愿的重要因素，在附件 1 中，123 家企业的信誉评级由银行专家评定为 A、B、C、D 四个等级，我们将其对应到 0、1、2、3 四个离散值。

5.2.5 是否违约量化

企业违约不能按时归还贷款是银行最为担心的信贷风险之一，因此企业会违约的概率是衡量信贷风险的标准，附件 1 已经明确给出了某一企业是否会违约，我们将其 $0-1$ 量化，将取值为“是”的数据量化为 1，将取值为“否”的数据量化为 0。

六、 问题一模型建立

银行贷款给企业后将会面临的信贷风险主要是企业存在不归还贷款的可能性，即企业有一定的违约概率，违约概率越大，企业不归还贷款的可能性越高，银行的信贷风险越大。而影响企业违约概率的因素有 5 个，即“企业实力”、“企业上游影响力”、“企业下游影响力”、“企业供求稳定性”、“信誉评级”, 为判断这 5 个因素的重要程度，我们利用决策树模型确定这 5 个因素的权重，并将归一化后的数据与权重组合输入到逻辑回归模型中得到一家企业的风险评分，逻辑回归模型得到的风险评分越大，企业违约可能性越大。

根据逻辑回归结果，确定一个阈值，如果结果大于这个阈值那么银行就不借款；如果结果小于这个阈值，银行再根据企业信誉评级、企业实力、客户流失率、坏账率等影响因素，在银行自身利润最大化，风险最小化的前提下决定借款的数量及年利率。具体流程图如图3所示。

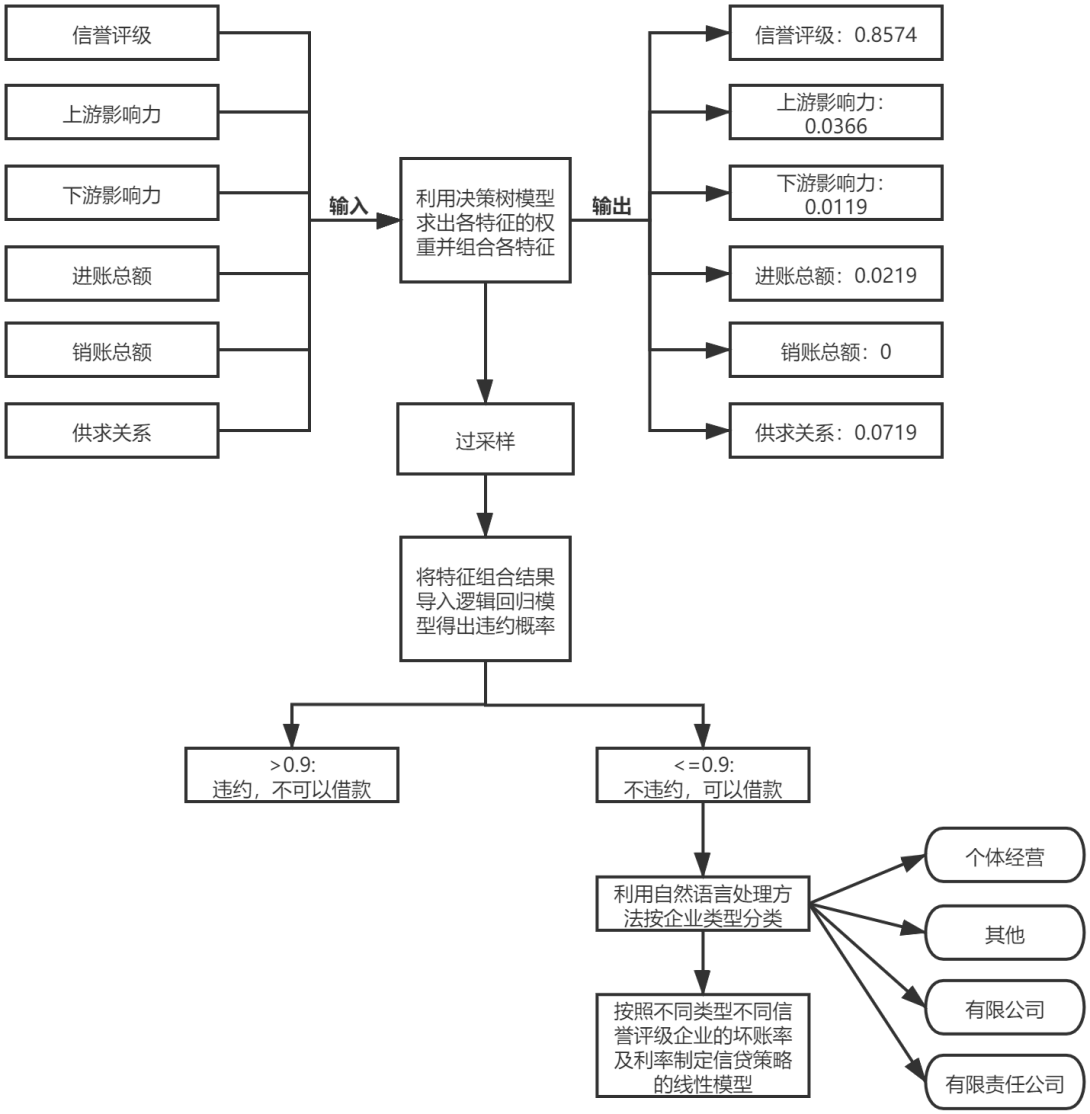

图 **3**　问题一路线图

6.1 利用决策树模型确定各特征的权重

首先建立决策树打分模型，将数据预处理后得到的 6 个特征变量中的“企业实力”、“企业上游影响力”、“企业下游影响力”、“企业供求稳定性”、“信誉等级”作为决策树的输入变量，将量化后的“是否违约”作为决策树的输出值。之后将 70% 的数据划分为训练集，30% 的数据划分为测试集，再进行决策树模型的训练与预测，并得到各个特征在划分违约情况时的权重。在最终训练结束后的模型中，其参数含义及取值如表3所示。

表 **3**　决策树参数表

参数名称	含义	取值
criterion	确定不纯度的计算方法，帮助找出最佳节点和最佳分枝，不纯度越低，决策数对训练集的拟合越好	基尼系数
random_state	设置分枝中的随机模式	30
splitter	控制决策树中的随机选项	best
max_depth	限制树的最大深度	6

在该模型的基础上我们达到了 0.9457 的准确率，为了验证模型的预测性能，避免过拟合我们进行了 15 折交叉验证，交叉验证结果如表4所示，其平均值为 0.9333，从结果中可以发现，该决策树模型的性能较为稳定，且分类结果较为准确。

表 **4**　决策树交叉验证结果

序号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
准确率	1	1	1	0.875	1	1	1	1	1	0.875	1	0.75	0.875	0.875	0.75

为进一步探究模型在测试集上的效果，我们绘制出了混淆矩阵，如图4所示，从图中可以看出有 35 家企业被正确地划分。有 2 家企业被错误地划分，这些错误会给银行带来难以估计的损失，尤其是会违约却被错误地划分到不会违约集合的企业中，因此要尽量避免这一错误。

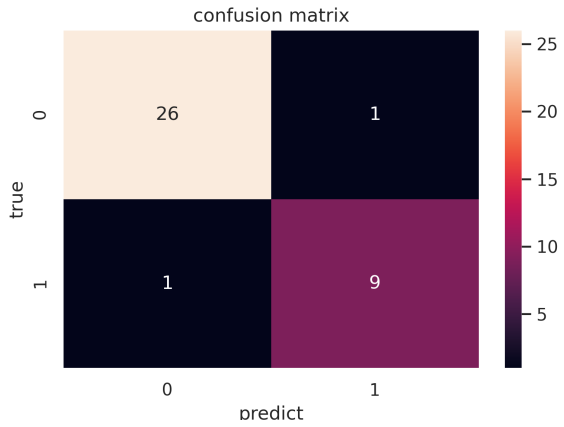

图 **4**　决策树混淆矩阵图

最终我们得出各个影响违约概率因素的权重，结果如表5所示。将该决策树分类结果可视化，其结果如图5所示, 其中“稳定标准差”就是指企业的“供求关系稳定性”。

表 **5**　各影响因素权重

	进项价税合计	销项价税合计	上游企业影响力
权重	0.0219	0.0000	0.0366
	下游企业影响力	稳定标准差	信誉等级
权重	0.0119	0.0719	0.8574

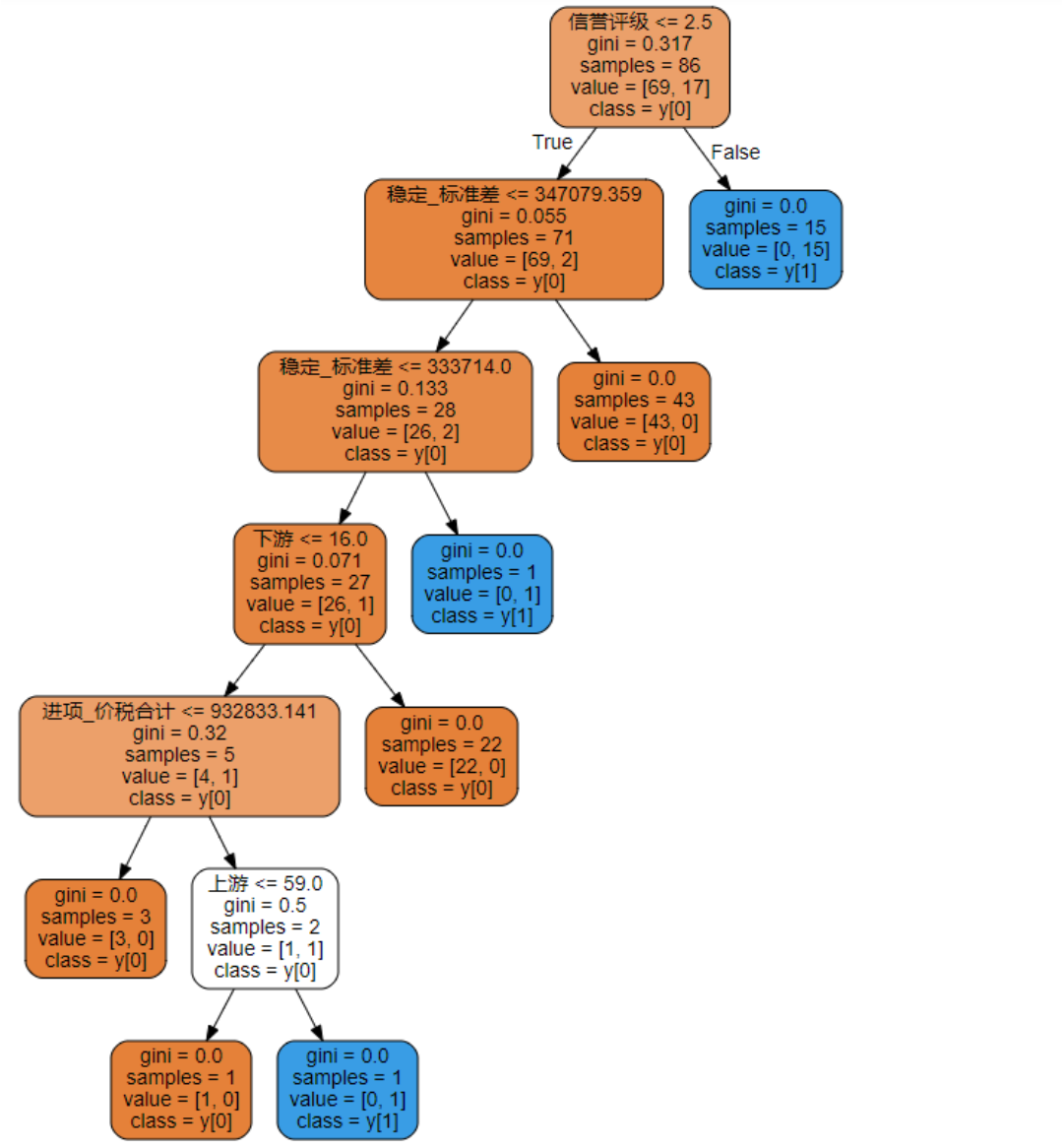

图 **5**　决策树分支图

从图5和表5中我们发现信誉评级的权重为 0.8574，说明它与企业的违约概率强相关，是具有决定性作用的特征，当企业的信誉评级量化得分 $<= 2.5$ 分，即信誉评级为 C 或 D 时，几乎可以认为这个企业会违约。因此银行需要慎重考虑是否借贷给信誉评级为 C 或 D 的企业，根据客观实际，结合企业的还款能力，部分信誉评级为 C 的企业可以获得银行信用贷款，但是所有信誉等级被评为 D 的企业均不能获得银行贷款。

6.2 利用逻辑回归模型求解各企业的违约概率

由决策树模型得出特征“进项价税合计”α_0，“销项价税合计”α_1，“上游企业影响力”β_0，“下游企业影响力”β_1，“企业供求稳定性标准差”σ，“信誉等级”γ 的相对权重 w 后，可以由此得到一个与权重有关的线性表达式：

$$z = 0.0219\alpha_0 + 0.0000\alpha_1 + 0.0366\beta_0 + 0.0119\beta_1 + 0.0719\sigma + 0.8574\gamma \tag{1}$$

将 z 作为中间变量代入下式中，可以利用逻辑回归模型预测出企业的信贷风险。

$$g(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

逻辑回归模型的输出是一个概率值，介于 0 和 1 之间，我们可以人为的设定一个阈值，作为二分类划分的界限，通常情况下，这个值设置为 0.5，但是对于风险预测这个具体的问题，结合常识和已知数据可知实际存在较大风险的企业是非常少的，属于类别不平衡的二分类问题。在 123 家企业数据集上反复调节参数后，可以确定阈值设置为 0.9 时，逻辑回归对这个二分类问题的预测程度最好，结果如图6所示，每一个企业的信贷风险评分如表6所示。

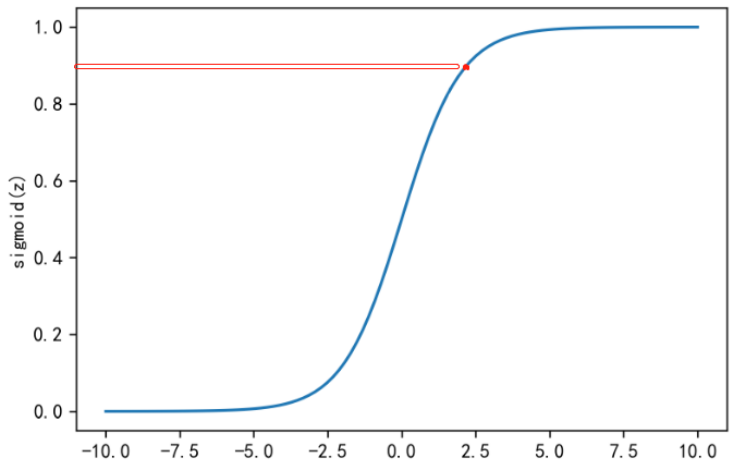

图 **6**　逻辑回归图

表 **6**　企业信贷风险评分表

	E1	E119	E46	E47	E48
信贷风险评分	0.5246	0.9290	0.8481	0.8481	0.5008

这些企业经过逻辑回归模型预测的输出值大部分集中在 1 的附近，其中得分超过 0.9 的被归为肯定会违约的一类，我们不给予贷款，低于 0.9 的企业被归为没有违约的一类，给予贷款。这样做既满足了对信誉评级为 D 的企业不予贷款，也将评级为 A, B, C 的企业中的有违约记录的识别出来，进而降低了风险。显然我们可以将逻辑回归的输出值作为一个企业的风险评分标准，得分越高，越容易违约。后续模型我们只需讨论关于这部分得分高的企业具体的信贷策略。

6.3 银行信贷策略的线性规划模型建立

银行作为盈利性机构, 它从事一切商业活动的根本性目的就是为了盈利，同时因为存在一定的信贷风险，所以银行在使得利润最大化的同时要尽量减少信贷风险。所以在构建银行信贷政策的线性规划模型时要考虑企业的违约概率（P）、信誉评级（γ）、企业实力、坏账率（V）、企业平均年利润（I）同时在指定贷款利率时银行也要考虑客户流失率（F），在可接受的客户流失范围内给出某一企业贷款利率。在这些因素中坏账率越高，信贷风险越大；客户流失率越高，银行收益越小；年利率越高，银行越容易盈利。

为了控制信贷风险并充分考虑企业的还款能力，在决定是否发放贷款时我们对根据逻辑回归模型计算出来的信贷风险评分高于 0.9 的企业、信誉评价被评定为 D 的企业、平均年利润低于 10 万元的企业不予放贷。

6.3.1 依据客户流失率确定不同信誉评级企业的贷款利率

由于不同的银行之间存在竞争，所以每一个银行都会有一个客户流失率，这是一个衡量银行流失顾客的概率，也是衡量企业在该银行贷款意愿的概率，对于每一个企业来说，他只有 (1– 客户流失率) 的概率选择在该银行贷款。

根据 TNS 发布的中国银行的调查报告，我国零售银行客户流失率最高，达到 30%，工商银行最低为 16%，借鉴该份报告我们认为银行顾客流失率的正常范围在 20% 与 30% 之间。结合银行的实际情况，信誉评级越低的企业贷款年利率越高，信誉评级越高的企业银行越不想流失。基于此，为降低模型求解难度，我们根据附件 3 中数据，确定每一个信誉评级的贷款年利率以及对应的客户流失率，具体取值结果如表7所示。

表 **7**　各信誉评价贷款年利率及客户流失率

	信誉评级 A	信誉评价 B	信誉评价 C
年利率	0.0505	0.0545	0.0585
客户流失率	0.2246	0.2768	0.2901

6.3.2 利用自然语言处理技术划分企业性质并计算坏账率

考虑到实际情况，会有一部分企业的贷款无法收回，其他的贷款可以完全收回，为了简化模型计算，将无法收回的那部分贷款亏损使用“坏账率”代替，将“坏账率“看作一个概率，赋予每个企业，对于每个企业来说，只有（1-坏账率）的贷款比例可以收回。

为了细化银行的信贷策略，我们在计算不同企业坏账率时不仅考虑企业的信誉评级，还考虑了企业的性质。我们利用自然语言处理技术，批量划分企业类型。首先建立一个企业性质分类文档，该文档存储了 4 个企业类型分别是：“个人经营”，“有限责任公司”，“有限公司”，“其他”。之后先利用正则表达式仅保留“企业名称”中的中文部分，之后利用 $jieba$ 库将“企业名称”分词，并将分词结果与企业性质分类文档中的 4 个性质的企业对比，将相同性质的企业归为同一类，如果分词结果中没有一个分词出现在企业性质文档中，则将该类企业分在“其他”类中。基于此，我们可以得到每一个企业的性质。

之后我们按照企业性质（4 类）和企业信誉评价（3 类）将企业划分为 12 个类别，根据附件 1 中 123 家企业的“是否违约”情况计算 12 个类别企业的坏账率，即同一类别里违约企业数的占比。结果如表8所示, 实际计算中我们用 0.00 来填补缺失值。

表 **8**　**12** 个类别企业的坏账率表

	个体经营	有限公司	有限责任公司	其他
信誉评级 A	Nan	0.00	0.00	0.00
信誉评级 B	1.00	0.00	0.00	0.00
信誉评级 C	0.00	0.05	0.14	0.00

6.3.3 银行信贷策略的线性模型

基于上述过程，我们得到如下银行信贷策略的线性模型，其中 V_{mn} 表示信誉评级为 m，企业类型为 n 的企业的坏账率，F_m 表示信誉评级为 m 时的客户流失率，R_m 表示

银行借款给信誉评级为 m 的企业时的利率，M_i 表示第 i 家企业获得的贷款，I_i 表示企业平均年利润，S 表示固定年度贷款总额度，以万元为单位。

$$\sum_{i=1}^{123}(1-V_{mn})(1-F_m)R_mM_i \tag{3}$$

$$\begin{cases} 10 <= M_i <= 100 & (4) \\ M_i <= I_i & (5) \\ \sum_{i=1}^{123} M_i <= S & (6) \end{cases}$$

在我们的信贷策略中，为控制信贷风险，我们首先决定对逻辑回归模型预测出的信贷风险得分在 0.9 以上的企业、信誉评级为 D 企业、平均年利润低于 10 万元的企业不予贷款。之后结合年利率、坏账率、客户流失率得到考虑了信贷风险的银行年利润收益公式（3），并把公式（3）的值最大化作为线性规划的目标。公式（4）、（5）、（6）分别是 3 个约束条件。公式（4）表示每一家企业可以获得的贷款额度在 10 万到 100 万的范围内；公式（5）表示每一家企业可以获得的贷款额度需要小于这一家企业的平均年利润，这一步是为了保证企业有还款能力；公式（6）表示银行给所有企业的贷款总金额不能超过它固定的年度信贷总额。通过该线性规划可以得到银行年利润最高时的各企业贷款金额。

七、 问题二求解

7.1 附件 2 数据处理

附件 2 与附件 1 相同特征的处理方式相同，但是附件 2 缺少“信誉评级”这一特征，所以为了更加精准的预测“信誉评级”的准确率，我们在问题 1 原有特征变量的基础上加入两个影响“信誉评级”的特征变量——“作废率”与“资金流通频次”。

7.1.1 作废率的量化

作废率表示某一企业作废发票数占总发票数的比率。将“销项发票信息”和“进项发票信息”中的“发票状态”取值为“作废发票”的数据按企业代号加总。并求出每个企业的发票总数，两者作商得到发票作废率。得出的部分企业作废率数据如表9所示。

表 **9**　部分企业发票作废率表

	E1	E4	E5	E119	E120	E123
作废率	0.0417	0.0757	0.0455	0.0730	0.3587	0.2461

7.1.2 资金流通频次的量化

资金流通频次表示公司活跃程度。将“销项发票信息”和“进项发票信息”中每个企业出现的数据个数加总求平均得到公司资金流通频次，得出的部分企业资金流通频次数据如表10所示。

表 **10**　部分企业资金流通频次表

	E1	E4	E5	E119	E120	E123
资金流通频次	81110	2231	1060	21	29	65

7.2 302 家企业的信誉评级预测

对于附件 2 中的 302 家无信誉评级记录的企业，我们对比了逻辑回归, 随机森林和线性支持向量机三种模型的预测能力，预测结果经交叉验证后的准确率可视化如图7所示，其中 $RandomForestClassifier$、$LinearSVC$、$LogisticRegression$ 分别指随机森林、支持向量机分类器、逻辑回归。由图7可以看出在三个模型的准确率对比中使用随机森林模型进行信誉评级预测效果最好，平均准确率可达到 70%。

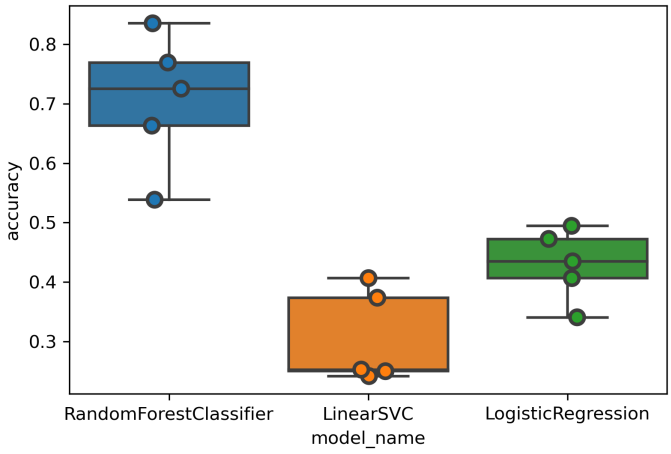

图 **7**　三个模型预测效果对比图

但是随机森林模型预测的平均准确率仍不是很高，因此我们对随机森林进行调参，调参结果如图8所示，当参数 ($n\_estimators$) 为 11 时，随机森林模型的预测效果准确率可达到 98%, 说明模型已经可以很准确的通过相应的特征变量预测正确信誉评级的类别。

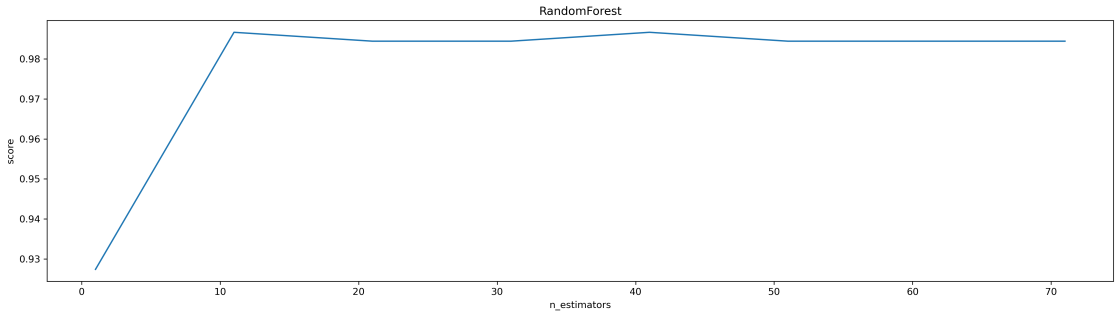

图 **8**　随机森林调参结果图

302 家企业的数据处理方法与附件 1 中的 123 家企业的处理方法相同。将 123 家企业的数据作为训练集，其特征“进项价税合计”,“销项价税合计”,“上游企业实力”,”下游企业实力”,“供求关系稳定程度”,”作废率”,“资金流通频次”作为模型的输入，经过调节参数，得到稳定的模型后，再将 302 家企业的数据作为预测数据输入模型中，得到 302 家企业的信誉评级，得到信誉评级后就可以通过问题 1 建立的决策树模型和逻辑回归模型计算企业的信贷风险评分等特征。

7.3 1 亿元年度信贷总额贷款策略

根据问题一中制定的策略，我们首先将三类共计 131 家企业排除（信贷风险评分高于 0.9、信誉评价为 D、平均年利润低于 10 万元），剩下的 171 家企业按照线性规划模型求解，该问题的线性方程模型如下。

$$\sum_{i=1}^{171}(1-V_{mn})(1-F_m)R_mM_i \tag{7}$$

$$\begin{cases} 10 <= M_i <= 100 & (8) \\ M_i <= I_i & (9) \\ \sum_{i=1}^{171} M_i <= 10000 & (10) \end{cases}$$

部分求解结果如下表11所示，在此分配情况下，银行的年收益为 427.5945 万元，具体 171 家企业的贷款金额分配情况见附件。

表 **11**　信贷策略部分表

企业代号	E126	E133	E278	E370
企业类别	个体经营	个体经营	有限公司	其他
信誉评级	A	B	B	C
坏账率	0	1	0	0
年利率	0.0505	0.0545	0.0545	0.0585
客户流失率	0.2246	0.2768	0.2768	0.2901
平均年利润（万元）	100229.7160	1923.5454	57.1199	15.1561
贷款额度（万元）	100.0000	10.0000	57.1199	15.1561

从模型结果可以看出，在我们的线性规划模型下，银行倾向于向企业实力强大即企业平均年利润高、信誉良好、坏账率低、年利率相对较高的企业发放较多的贷款；向信誉良好、坏账率低但企业实力相对较弱的企业发放较少的贷款；对信誉评级低、坏账率高的企业发放最低额度贷款。这与银行实际吻合，进一步证明了我们信贷策略线性规划模型的正确性。

八、 问题三：考虑发生突发状况时银行的信贷策略调整

8.1 构建滑动平均动量监控模型监控突发因素

现实生活中企业可能会受到突发因素影响，导致其生产经营和经济效益发生较大幅度的波动，并且不同行业，不同类别的企业所受的影响也会不同，因此我们构建了一个滑动平均动量监控模型来监测企业是否受到突发因素的影响。

企业所受的突发因素影响以新冠病毒为例，首先建立一个监测变量 m_t，根据 m_t 的变动幅度来确定疫情对企业是否有显著影响。m_t 的数学表达式如下，t 为月份，m_t 为 t 月份的历史累计利润，g_t 为企业的月盈利金额，θ_0 为一个常数，根据经验一般 θ_0 取 0.9。

$$m_t = \theta_0 m_{t-1} + (1-\theta_0)\, g_t \tag{11}$$

为了避免偏差，将 m_t 进行无偏估计，得到如下数学表达式:

$$\hat{m_t} = \frac{m_t}{1-\theta^t} \tag{12}$$

当 $|\Delta m_i - \frac{\sum_1^{i-1}\Delta m_i}{i-1} < \varepsilon|$ 时，说明 m_t 未监测到异常，企业利润稳定，其不受新冠疫情影响，并且银行对此公司的信贷策略不发生改变，当 $|\Delta m_i - \frac{\sum_1^{i-1}\Delta m_i}{i-1} > \varepsilon|$ 时认为 m_t 监测到异常，企业利润发生波动，其受到了新冠疫情的影响。

以代号为 $E269$ 的公司为例，进行滑动平均动量监控，如图 9所示，起初有一个较大幅度的攀升是因为历史数据较少，模型需要热启动，启动平稳后模型可正常监控，监控过程中曲线基本围绕 0 上下波动，其中 2017 – 9 月和 11 月有较大幅度的波动，说明此企业在 2017 年 9 月和 11 月受到了突发因素的影响，9 月时受突发因素影响利润变大，11 月时受突发因素影响利润减少。

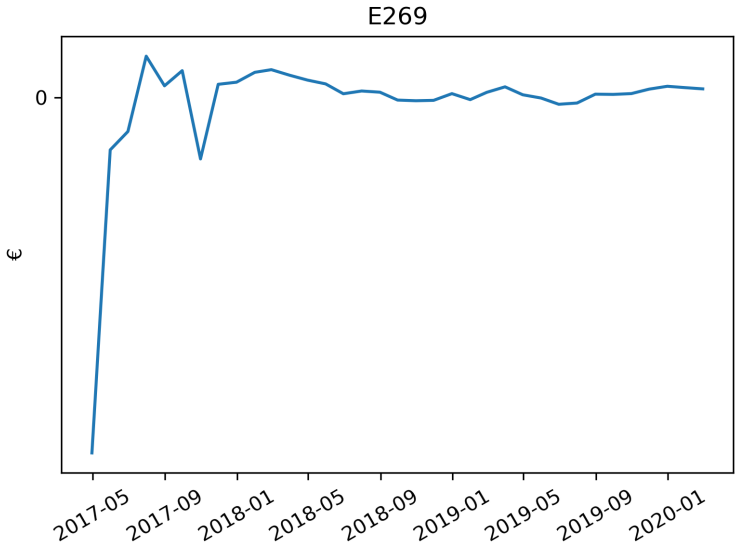

图 **9**　滑动平均监测图

m_t 在监测到异常的情况下，为了确定银行对企业信贷策略的调整计划，利用自然语言处理对“企业名称”进行行业分类，共分为 6 类，分别为：建筑业，制造业，服务业，交通运输业，信息技术业和其他。再将“企业行业归属”与问题 2 得出的“企业类型”作为两个新的特征加入到问题 2 原有的决策树模型中，训练出各特征新的特征权重，所得的新特征权重见表 12, 并利用逻辑回归模型得出信贷风险评分，所得的信贷风险评分见表 13。根据信贷风险评分数据，银行判断是否对企业借贷进行信贷策略调整，以此为基础，根据问题 2 的线性规划模型具体调整银行对企业的贷款额度与年利率。

从表13 中可以看出，各个特征对信贷风险评分的影响程度较为相近，仅“企业类别”和“行业类别”权重相对较大。且相较于问题一问题二各个影响因素的权重都发生变化，以至于新冠疫情爆发后企业信贷风险评分发生较大变化，所以银行可以据此利用信贷策略的线性规划模型改变信贷策略。

表 **12**　特征权重

	进项价税合计	销项价税合计	上游企业影响力	下游企业影响力
权重	0.1487	0.1290	0.2907	0.1141
	平均年利润	行业类型	企业类别	资金流通频次
权重	0.1405	0.2818	0.3119	0.2906

表 **13**　企业信贷风险评分表

	E129	E153	E193	E214	E290
信贷风险评分	0.8447	0.5624	0.9866	0.5138	0.4759

8.2 银行信贷新策略

同问题二中确定银行信贷策略的过程，我们在这题也采用问题一中的线性规划策略。由于突发新冠疫情主要影响企业的还款能力即增加了银行的信贷风险，所有显然会有一批原本可以获得银行贷款的企业无法获得贷款。根据问题一的策略，我们首先排除三类高风险的企业即信用评分高于 0.9、信誉评价为 D、平均年利润低于 10 万元的企业。这三类企业共计 147 家，显然高于问题二的 131 家。对于剩下的 155 家企业，我们按照线性规划模型求解，方程如下：

$$\sum_{i=1}^{155}(1-V_{mn})(1-F_m)R_mM_i \tag{13}$$

$$\begin{cases} 10 <= M_i <= 100 & (14) \\ M_i <= I_i & (15) \\ \sum_{i=1}^{155} M_i <= 10000 & (16) \end{cases}$$

为了进行新冠肺炎前后的企业贷款对比，我们选取与问题二中企业代码相同的企业做对比，他们的贷款情况如下所示：

表 **14**　新信贷策略部分表

企业代号	E126	E133	E278	E370
企业类别	个体经营	个体经营	有限公司	其他
信誉评级	A	B	B	C
坏账率	0	1	0	0
年利率	0.0505	0.0545	0.0545	0.0585
客户流失率	0.2246	0.2768	0.2768	0.2901
平均年利润（万元）	100229.7160	1923.5454	57.1199	15.1561
贷款额度（万元）	100.0000	10.0000	24.3675	12.3647

我们可以看出，对于实力强大、信誉良好的企业，银行依旧愿意给予他一个较大的贷款金额，但是对于实力较弱或者信誉不良的企业银行会选择降低给予他们的贷款。

九、 模型优缺点

优点：从决策树到逻辑回归再到线性模型，每个模型负责处理一小部分任务，各司其职，其中结构清晰完整，在此基础上，第二问加入随机森林模型可预测信誉评价等级。第三问加入滑动平均动量监测模型对突发因素进行监控进而决定银行是否改变信贷策略。整体上条件清晰，可扩展性强。

缺点：由于数据量少，我们的各项指标都用相似指标替代，造成了一定的误差，求解线性模型时，为了减少计算量将利率根据客户流失率和信誉评级固定下来，可能与最优结果产生偏差。

参考文献

[1] 张琳，张琪. 新冠肺炎疫情影响下金融支持实体经济的策略研究 [J/OL]. 边疆经济与文化,2020 -6.

[2] 胡腾. 城市商业银行中小企业信贷策略研究——以郑州银行为例 [D]. 山东:10422,2012.

[3] 龙建辉. 新冠肺炎疫情下广东中小企业应对策略 [J]. 广东经济,2020,4(16):1-23.

附录 A　支撑材料说明目录

文件名	文件内容
代码	存放所有的代码程序，格式为.py
问题一中间数据	存放问题一数据预处理及后期建模产生的所有数据，比如量化后的企业实力
问题二中间数据	存放问题二数据预处理及后期建模产生的所有数据，比如作废率
问题三结果数据	存放解决问题三信贷策略问题所需的所有数据，以及信贷策略线性规划结果

附录 B　数据概览

```
import pandas as pd

企业信息1 = pd.read_excel(r'附件1：123家有信贷记录企业的相关数据.xlsx',sheet_name=0)
企业信息1

进项发票1 = pd.read_excel(r'附件1：123家有信贷记录企业的相关数据.xlsx',sheet_name=1)
进项发票1

销项发票1 = pd.read_excel(r'附件1：123家有信贷记录企业的相关数据.xlsx',sheet_name=2)
销项发票1

企业信息2 = pd.read_excel(r'附件2：302家无信贷记录企业的相关数据.xlsx',sheet_name=0)
企业信息2

销项发票2 = pd.read_excel(r'附件2：302家无信贷记录企业的相关数据.xlsx',sheet_name=1)
销项发票2

进项发票2 = pd.read_excel(r'附件2：302家无信贷记录企业的相关数据.xlsx',sheet_name=2)
进项发票2

年利率与客户流失率 = pd.read_excel(r'附件3：银行贷款年利率与客户流失率关系的统计数据.xlsx')
年利率与客户流失率
```

附录 C　数据预处理

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']

企业信息1 = pd.read_excel(r'附件1：123家有信贷记录企业的相关数据.xlsx',sheet_name=0)
进项发票1 = pd.read_excel(r'附件1：123家有信贷记录企业的相关数据.xlsx',sheet_name=1)
销项发票1 = pd.read_excel(r'附件1：123家有信贷记录企业的相关数据.xlsx',sheet_name=2)

# 去除无效发票
进项发票1 = 进项发票1[进项发票1['发票状态']=='有效发票']
销项发票1 = 销项发票1[销项发票1['发票状态']=='有效发票']
销项发票1 = 销项发票1[销项发票1['发票状态']=='有效发票']

上游=进项发票1[['企业代号','销方单位代号']]
下游=销项发票1[['企业代号','购方单位代号']]

from collections import Counter
Counter(进项发票1['发票状态'])


# ## 上下游


上游.groupby('企业代号').nunique()[['销方单位代号']].to_csv('./task1/上游.csv',encoding='gbk')


下游.groupby('企业代号').nunique()[['购方单位代号']].to_csv('./task1/下游.csv',encoding='gbk')


# ## 企业实力

from collections import Counter
Counter(进项发票1['发票状态'])

进项实力 = 进项发票1[进项发票1['发票状态']=='有效发票']
进项实力_价税合计 = 进项实力[['企业代号','价税合计']].groupby(' 企业代号').sum()
进项实力_价税合计 = 进项实力_价税合计.rename(columns={'价税合计':'进项_价税合计'})
销项实力 = 销项发票1[销项发票1['发票状态']=='有效发票']
销项实力_价税合计 = 销项实力[['企业代号','价税合计']].groupby(' 企业代号').sum()
销项实力_价税合计 = 销项实力_价税合计.rename(columns={'价税合计':'销项_价税合计'})
销项实力_价税合计
实力无归一化 = pd.concat([进项实力_价税合计,销项实力_价税合计],axis=1)
实力无归一化.to_csv('./task1/实力无归一化.csv',encoding='gbk')
实力无归一化

sd = 进项实力_价税合计.describe()
进项_价税合计_归一化 = 进项实力_价税合计.apply(lambda x:
    (x-sd.loc['min'].values.item())/(sd.loc['max'].values.item()-sd.loc['min'].values.item()))
```

```
进项_价税合计_标准化 = 进项实力_价税合计.apply(lambda x:
    (x-sd.loc['mean'].values.item())/sd.loc['std'].values.item())
进项_价税合计_归一化=进项_价税合计_归一化.rename(columns={'进项_价税合计':'进项_价税合计_归一化'})
进项_价税合计_标准化=进项_价税合计_标准化.rename(columns={'进项_价税合计':'进项_价税合计_标准化'})

sd2 = 销项实力_价税合计.describe()
销项_价税合计_归一化 = 销项实力_价税合计.apply(lambda x:
    (x-sd.loc['min'].values.item())/(sd.loc['max'].values.item()-sd.loc['min'].values.item()))
销项_价税合计_标准化 = 销项实力_价税合计.apply(lambda x:
    (x-sd.loc['mean'].values.item())/sd.loc['std'].values.item())
销项_价税合计_归一化=销项_价税合计_归一化.rename(columns={'销项_价税合计':'销项_价税合计_归一化'})
销项_价税合计_标准化=销项_价税合计_标准化.rename(columns={'销项_价税合计':'销项_价税合计_标准化'})

实力归一化 =
pd.concat([进项_价税合计_归一化,进项_价税合计_标准化,销项_价税合计_标准化,销项_价税合计_归一化],axis=1)
实力归一化.to_csv('./task1/实力归一化.csv',encoding='gbk')


# ## 供求关系以E1为例

Counter(进项发票1['企业代号'])

df = []
for M in 企业信息1['企业代号'].values:
    M企业 = 进项发票1[进项发票1['企业代号']==M][['开票日期','价税合计']]
    进项价税合计 = M企业.groupby('开票日期').sum()
    进项价税合计 = 进项价税合计.rename(columns={'价税合计':'进项价税合计'})
    M企业2 = 销项发票1[销项发票1['企业代号']==M][['开票日期','价税合计']]
    销项价税合计 = M企业2.groupby('开票日期').sum()
    销项价税合计 = 销项价税合计.rename(columns={'价税合计':'销项价税合计'})
    供求关系 = pd.concat([进项价税合计,销项价税合计],axis=1)
    供求关系 = 供求关系.fillna(0)
    供求关系['差']=供求关系['销项价税合计']-供求关系['进项价税合计']

    M企业['year'], M企业['month'], M企业['day'] = list(zip(*M企业['开票日期'].apply
(lambda d: (d.year, d.month, d.day))))
    一七年价税合计 =
        M企业[M企业['year']==2017][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2017年价税合计'})
    一八年价税合计 =
        M企业[M企业['year']==2018][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2018年价税合计'})
    一九年价税合计 =
        M企业[M企业['year']==2019][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2019年价税合计'})
    二零年价税合计 =
        M企业[M企业['year']==2020][['价税合计','month']].groupby('month').sum().rename(columns=
```

```
{'价税合计':'2020年价税合计'})

    M价税合计 = pd.concat([一七年价税合计,一把年价税合计,一久年价税合计,二零年价税合计],axis=1)
    M价税合计 = M价税合计.fillna(0)
    # M价税合计

    M企业2['year'], M企业2['month'], M企业2['day'] = list(zip(*M企业2['开票日期'].apply(lambda
        d: (d.year, d.month, d.day))))
    一七年价税合计2 =
        M企业2[M企业2['year']==2017][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2017年价税合计'})
    一八年价税合计2 =
        M企业2[M企业2['year']==2018][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2018年价税合计'})
    一九年价税合计2 =
        M企业2[M企业2['year']==2019][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2019年价税合计'})
    二零年价税合计2 =
        M企业2[M企业2['year']==2020][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2020年价税合计'})

    M价税合计2 =
        pd.concat([一七年价税合计2,一八年价税合计2,一九年价税合计2,二零年价税合计2],axis=1)
    M价税合计2 = M价税合计2.fillna(0)
    # M价税合计2

    a = M价税合计2['2017年价税合计']-M价税合计['2017年价税合计']
    b = M价税合计2['2018年价税合计']-M价税合计['2018年价税合计']
    c = M价税合计2['2019年价税合计']-M价税合计['2019年价税合计']
    d = M价税合计2['2020年价税合计']-M价税合计['2020年价税合计']

    M利润 = pd.concat([a,b,c,d],axis=1)
    M利润= M利润.rename(columns={'2017年价税合计':'2017利润','2018年价税合计':'2018
        利润','2019年价税合计':'2019利润','2020年价税合计':'2020利润'})
    df.append(M利润)

df[99].plot()

m = np.array(df)
np.save('./task1/all_plot.npy',m)


# # ================== task2

企业信息2 = pd.read_excel(r'附件2: 302家无信贷记录企业的相关数据.xlsx',sheet_name=0)
销项发票2 = pd.read_excel(r'附件2: 302家无信贷记录企业的相关数据.xlsx',sheet_name=1)
进项发票2 = pd.read_excel(r'附件2: 302家无信贷记录企业的相关数据.xlsx',sheet_name=2)
```

```
## 去除无效发票
进项发票2 = 进项发票2[进项发票2['发票状态']=='有效发票']
销项发票2 = 销项发票2[销项发票2['发票状态']=='有效发票']

上游=进项发票2[['企业代号','销方单位代号']]
下游=销项发票2[['企业代号','购方单位代号']]


# ## 上下游

上游 = 上游.groupby('企业代号').nunique()[['销方单位代号']]
下游 = 下游.groupby('企业代号').nunique()[['购方单位代号']]
上下游 = pd.concat([上游,下游],axis=1)
上下游 = 上下游.rename(columns={'销方单位代号':'上游','购方单位代号':'下游'})
上下游.to_csv('./task2/上下游无归一化.csv',encoding='gbk')
上下游['上游']=上下游['上游'].apply(lambda x:(x-上下游.describe()
.loc['min','上游'])/(上下游.describe().loc['max','上游']
-上下游.describe().loc['min','上游']))
上下游['下游']=上下游['下游'].apply(lambda x:(x-上下游.describe()
.loc['min','下游'])/(上下游.describe().loc['max','下游']
-上下游.describe().loc['min','下游']))
上下游.to_csv('./task2/上下游.csv',encoding='gbk')


# ## 企业实力

进项实力 = 进项发票2[进项发票2['发票状态']=='有效发票']
进项实力_价税合计 = 进项实力[['企业代号','价税合计']].groupby(' 企业代号').sum()
# 进项实力_价税合计

sd = 进项实力_价税合计.describe()
进项_价税合计_归一化 = 进项实力_价税合计.apply(lambda x:
    (x-sd.loc['min'].values.item())/(sd.loc['max'].values.item()-sd.loc['min'].values.item()))
进项_价税合计_标准化 = 进项实力_价税合计.apply(lambda x:
    (x-sd.loc['mean'].values.item())/sd.loc['std'].values.item())
进项_价税合计_归一化=进项_价税合计_归一化.rename(columns={'价税合计':'进项_价税合计_归一化'})
进项_价税合计_标准化=进项_价税合计_标准化.rename(columns={'价税合计':'进项_价税合计_标准化'})


销项实力 = 销项发票2[销项发票2['发票状态']=='有效发票']
销项实力_价税合计 = 销项实力[['企业代号','价税合计']].groupby(' 企业代号').sum()
# 销项实力_价税合计

sd2 = 销项实力_价税合计.describe()
销项_价税合计_归一化 = 销项实力_价税合计.apply(lambda x:
    (x-sd.loc['min'].values.item())/(sd.loc['max'].values.item()-sd.loc['min'].values.item()))
```

```
销项_价税合计_标准化 = 销项实力_价税合计.apply(lambda x:
    (x-sd.loc['mean'].values.item())/sd.loc['std'].values.item())
销项_价税合计_归一化=销项_价税合计_归一化.rename(columns={'价税合计':'销项_价税合计_归一化'})
销项_价税合计_标准化=销项_价税合计_标准化.rename(columns={'价税合计':'销项_价税合计_标准化'})
#
实力无标准化 = pd.concat([进项实力_价税合计,销项实力_价税合计],axis=1)
实力无标准化.to_csv('./task2/实力无标准化.csv',encoding='gbk')
实力 =
    pd.concat([进项_价税合计_归一化,进项_价税合计_标准化,销项_价税合计_标准化,销项_价税合计_归一化]
,axis=1)
实力.to_csv('./task2/实力.csv',encoding='gbk')


# ## 供求关系

df = []
for M in 企业信息2['企业代号'].values:
    M企业 = 进项发票2[进项发票2['企业代号']==M][['开票日期','价税合计']]
    进项价税合计 = M企业.groupby('开票日期').sum()
    进项价税合计 = 进项价税合计.rename(columns={'价税合计':'进项价税合计'})
    M企业2 = 销项发票2[销项发票2['企业代号']==M][['开票日期','价税合计']]
    销项价税合计 = M企业2.groupby('开票日期').sum()
    销项价税合计 = 销项价税合计.rename(columns={'价税合计':'销项价税合计'})
    供求关系 = pd.concat([进项价税合计,销项价税合计],axis=1)
    供求关系 = 供求关系.fillna(0)
    供求关系['差']=供求关系['销项价税合计']-供求关系['进项价税合计']

    M企业['year'], M企业['month'], M企业['day'] = list(zip(*M企业['开票日期'].apply(lambda d:
        (d.year, d.month, d.day))))
    一七年价税合计 =
        M企业[M企业['year']==2017][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2017年价税合计'})
    一八年价税合计 =
        M企业[M企业['year']==2018][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2018年价税合计'})
    一九年价税合计 =
        M企业[M企业['year']==2019][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2019年价税合计'})
    二零年价税合计 =
        M企业[M企业['year']==2020][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2020年价税合计'})

    M价税合计 = pd.concat([一七年价税合计,一八年价税合计,一九年价税合计,二零年价税合计],axis=1)
    M价税合计 = M价税合计.fillna(0)
    # M价税合计
```

```
    M企业2['year'], M企业2['month'], M企业2['day'] = list(zip(*M企业2['开票日期'].apply(lambda
        d: (d.year, d.month, d.day))))
    一七年价税合计2 =
        M企业2[M企业2['year']==2017][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2017年价税合计'})
    一八年价税合计2 =
        M企业2[M企业2['year']==2018][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2018年价税合计'})
    一九年价税合计2 =
        M企业2[M企业2['year']==2019][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2019年价税合计'})
    二零年价税合计2 =
        M企业2[M企业2['year']==2020][['价税合计','month']].groupby('month').sum().rename(columns=
{'价税合计':'2020年价税合计'})

    M价税合计2 =
        pd.concat([一七年价税合计2,一八年价税合计2,一九年价税合计2,二零年价税合计2],axis=1)
    M价税合计2 = M价税合计2.fillna(0)
    # M价税合计2

    a = M价税合计2['2017年价税合计']-M价税合计['2017年价税合计']
    b = M价税合计2['2018年价税合计']-M价税合计['2018年价税合计']
    c = M价税合计2['2019年价税合计']-M价税合计['2019年价税合计']
    d = M价税合计2['2020年价税合计']-M价税合计['2020年价税合计']

    M利润 = pd.concat([a,b,c,d],axis=1)
    M利润= M利润.rename(columns={'2017年价税合计':'2017利润','2018年价税合计':'2018
        利润','2019年价税合计':'2019利润','2020年价税合计':'2020利润'})
    df.append(M利润)

m = np.array(df)
np.save('./task2/all_plot.npy',m)
```

附录 D　决策树 + 逻辑回归

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus']=False
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.model_selection import train_test_split, cross_val_score
```

```
data1 = pd.read_csv('./task1/data1无归一化.csv',encoding='gbk',index_col=0)
作废率_频次 = pd.read_csv('./task1/作废率_频次.csv',encoding='gbk').set_index('企业代号')
# sd = data1.describe()
# data1['信誉评级'] = data1['违约率'].apply(lambda
    x:(x-sd.loc['min','违约率'])/(sd.loc['max','违约率']-sd.loc['min','违约率']))
data1 = pd.concat([data1,作废率_频次],axis=1)
data1.to_csv('./task1/features_labels.csv',encoding='gbk')
data1

X = data1[['进项_价税合计','销项_价税合计','上游','下游','稳定_
    标准差','信誉评级','作废率','资金流通频次']]
y = data1['是否违约']

train_X, val_X, train_y, val_y = train_test_split(X.to_numpy(), y, test_size=0.3,
    random_state=0)
dt = DecisionTreeClassifier()
dt.fit(train_X, train_y)
dt.score(val_X, val_y)
cross_val_score(dt, X, y, cv=15)

dt.feature_importances_

import graphviz
import os
os.environ["PATH"] += os.pathsep + r'C:\Users\chenhongda\Downloads\Graphviz\bin'

dot_data = tree.export_graphviz(dt,
                            feature_names = X.iloc[:,:6].columns, class_names = True,
                            filled = True, rounded = True)
graph = graphviz.Source(dot_data)
graph

dat = data1[['进项_价税合计','销项_价税合计','上游','下游','稳定_标准差','信誉评级']]
dat

# logistic regression

def sigmoid(x):
    w = np.array([0.02199488, 0.,0.03665814, 0.01194784, 0.07190121, 0.85749793])
    z = np.dot(w,x)
    return 1/(1+np.exp(-z))

data1 = pd.read_csv('./task1/data1归一化.csv',encoding='gbk',index_col=0)
X = data1[['进项_价税合计_归一化','销项_价税合计_归一化','上游','下游','稳定_标准差','信誉评级']]
y = data1['是否违约']

xx=[]
```

```
yy=[]
for i in X.to_numpy():
    xx.append(i)
    yy.append(sigmoid(i))
plt.plot(xx,yy)

dat['风险评分'] = yy
dat[['风险评分']].to_csv('./风险评分.csv',encoding='gbk')
yy

yyy=[]
for i in yy:
    if i > 0.9:
        yyy.append(1)
    else:
        yyy.append(0)
yyy

from collections import Counter
Counter([y[i]==yyy[i] for i in range(len(y))])

y

def sig(x):
    return 1/(1+np.exp(-x))
x = np.linspace(-10,10,100)
y = sig(x)

plt.plot(x,y)
plt.xlabel('z')
plt.ylabel('sigmoid(z)')
plt.savefig('./sigmoid.png',dpi=300)
plt.show()
```

附录 E　预测信誉评级

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score


实力无归一化 = pd.read_csv('./task1/实力无归一化.csv',encoding='gbk').set_index('企业代号')
```

```
实力无归一化

上游 = pd.read_csv('./task1/上游.csv',encoding='gbk').set_index('企业代号')
下游 = pd.read_csv('./task1/下游.csv',encoding='gbk').set_index('企业代号')
上下游 = pd.concat([上游,下游],axis=1)
上下游 = 上下游.rename(columns={'销方单位代号':'上游','购方单位代号':'下游'})

上下游

企业信息1 = pd.read_excel(r'附件1：123家有信贷记录企业的相关数据.xlsx',sheet_name=0)
企业信息1 = 企业信息1.set_index('企业代号')
企业信息1

df = np.load('./task1/all_plot.npy',allow_pickle=True)
dff = []
for i in df:
    dff.append(i.describe().loc['std'].mean())

企业信息1['稳定_标准差'] = dff
企业信息1 = 企业信息1.fillna(0)
企业信息1

def tran(x):
    if x=='是':
        return 1
    else:
        return 0

企业信息1['是否违约'] = 企业信息1['是否违约'].apply(tran)
A = 企业信息1[企业信息1['信誉评级']=='A']['是否违约'].sum()/企业信息1[企业信息1['信誉评级']=='A']
['是否违约'].count()
B = 企业信息1[企业信息1['信誉评级']=='B']['是否违约'].sum()/企业信息1[企业信息1['信誉评级']=='B']
['是否违约'].count()
C = 企业信息1[企业信息1['信誉评级']=='C']['是否违约'].sum()/企业信息1[企业信息1['信誉评级']=='C']
['是否违约'].count()
D = 企业信息1[企业信息1['信誉评级']=='D']['是否违约'].sum()/企业信息1[企业信息1['信誉评级']=='D']
['是否违约'].count()

def tranf(x):
    if x=='A':
        return 0
    elif x=='B':
        return 1
    elif x=='C':
        return 2
    elif x=='D':
        return 3
```

```
企业信息1['信誉评级'] = 企业信息1['信誉评级'].apply(tranf)
企业信息1

data1无归一化 = pd.concat([实力无归一化,上下游,企业信息1],axis=1)
data1无归一化.to_csv('./task1/data1无归一化.csv',encoding='gbk')

# 归一化
上下游['上游']=上下游['上游'].apply(lambda
    x:(x-上下游.describe().loc['min','上游'])/(上下游.describe().loc['max','上游']
-上下游.describe().loc['min','上游']))
上下游['下游']=上下游['下游'].apply(lambda
    x:(x-上下游.describe().loc['min','下游'])/(上下游.describe().loc['max','下游']
-上下游.describe().loc['min','下游']))

实力归一化 = pd.read_csv('./task1/实力归一化.csv',encoding='gbk').set_index('企业代号')

企业信息1['稳定_标准差'] = 企业信息1['稳定_标准差'].apply(lambda
    x:(x-企业信息1['稳定_标准差'].describe().loc['min'])/(企业信息1['稳定_标准差']
.describe().loc['max']- 企业信息1['稳定_标准差'].describe().loc['min']))

data1 = pd.concat([实力归一化,上下游,企业信息1],axis=1)
data1.to_csv('./task1/data1归一化.csv',encoding='gbk')

# ## model

# In[221]:


X = data1无归一化[['进项_价税合计','销项_价税合计','上游','下游','稳定_标准差']].to_numpy()
y = data1无归一化['违约率']
display(X,y)

from imblearn.over_sampling import SMOTE
from collections import Counter
smo = SMOTE(random_state=42)
X_smo, y_smo = smo.fit_sample(X,y)

len(X_smo)

train_X, val_X, train_y, val_y = train_test_split(X_smo, y_smo, test_size=0.3, random_state=0)

from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
knn = KNeighborsClassifier(n_neighbors=45, weights='distance')
clf = RandomForestClassifier(n_estimators=45)
knn.fit(train_X, train_y)
acc1 = knn.score(val_X, val_y)
```

```
clf.fit(train_X,train_y)
acc1=clf.score(val_X, val_y)
print(acc1)
pickle.dump(lr, open("./task1/model_knn.pkl", "wb"))

acc = []
for k in range(5, 100):
    forest = RandomForestClassifier(n_estimators=k)
    score = cross_val_score(forest, X, y, cv=10, scoring="accuracy").mean()
    acc.append(score)
print('最优score: ', max(acc), '最优n_estimator:', ([*range(5, 100)][acc.index(max(acc))]))
plt.plot(range(5, 100), acc)
plt.xlabel("decision_tree_number")
plt.ylabel("accuracy")
plt.title("RandomForest")
plt.show()

# logistic regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=1e9)
lr.fit(train_X, train_y)
lr.score(val_X, val_y)
# cross_val_score(forest, X_smo, y_smo, cv=10, scoring="accuracy").mean()

from sklearn.metrics import confusion_matrix as CM
y_pred=clf.predict(val_X)
CM(val_y,y_pred)

import pickle

from collections import Counter
Counter(企业信息1['信誉评级'])

B

C



# # ==================== task2

实力 = pd.read_csv('./task2/实力.csv',encoding='gbk').set_index('企业代号')
上下游 = pd.read_csv('./task2/上下游.csv',encoding='gbk')
企业信息2 =
    pd.read_excel(r'附件2: 302家无信贷记录企业的相关数据.xlsx',sheet_name=0).set_index('企业代号')
实力

上下游 = 上下游.set_index('企业代号')
```

```
上下游

df = np.load('./task2/all_plot.npy',allow_pickle=True)
dff = []
for i in df:
    dff.append(i.describe().loc['std'].mean())
企业信息2['稳定_标准差'] = dff
企业信息2 = 企业信息2.fillna(0)
企业信息2['稳定_标准差'] = 企业信息2['稳定_标准差'].apply(lambda
    x:(x-企业信息2['稳定_标准差'].describe().loc['mean'])/企业信息2['稳定_标准差'].describe().loc['std'])
企业信息2

data2 = pd.concat([实力,上下游,企业信息2],axis=1)
信息2 = pd.read_csv('./task2/企业信息2.csv',encoding='gbk',index_col=0).set_index(' 企业代号')
待预测_归一化 = pd.concat([data2,信息2],axis=1)
待预测_归一化.to_csv('./task2/待预测_归一化.csv',encoding='gbk')

X =
    待预测_归一化[['进项_价税合计_归一化','销项_价税合计_归一化','上游','下游','稳定_标准差']].to_numpy()
model = pickle.load(open("./task1/model_randomForest.pkl", "rb"))
y = model.predict(X)
y

def tran(x):
    if x == 0:
        return 'A'
    elif x == 1:
        return 'B'
    elif x == 2:
        return 'C'
    elif x == 3:
        return 'D'
待预测_归一化['信誉评级预测'] = y


待预测_归一化['信誉评级预测'] = 待预测_归一化['信誉评级预测'].apply(tran)
待预测_归一化

待预测_归一化.to_csv('./task2/预测后.csv',encoding='gbk')
待预测_归一化 = pd.read_csv('./task2/预测后.csv',encoding='gbk')
待预测_归一化

年利率与客户流失率 = pd.read_excel(r'附件3: 银行贷款年利率与客户流失率关系的统计数据.xlsx')
年利率与客户流失率 = 年利率与客户流失率[2:]
年利率与客户流失率

0 = 待预测_归一化[['信誉评级预测','企业类别']]
```

```
O['信誉评级预测']

def f(v,F,R,M):
   return (1-v)*(1-F)*R*M

def bad_budget(level,cate):
    if level == 'A' and cate == '个体经营':
        return 1
    elif level == 'B' and cate == '个体经营':
        return 1
    elif level == 'C' and cate == '有限公司':
        return 0.05
    elif level == 'C' and cate == '有限责任公司':
        return 0.1428
    else:
        return 0

def R_F(R,df,o,oo):
    if o.loc[oo,'信誉评级预测'] == 'A':
        return df.loc[R,'客户流失率']
    elif o.loc[oo,'信誉评级预测'] == 'B':
        return df.loc[R,'Unnamed: 2']
    elif o.loc[oo,'信誉评级预测'] == 'C':
        return df.loc[R,'Unnamed: 3']

利润={}
I = 0
for M in range(100,1000):
    for R in np.arange(0.0425,0.15,0.004):
        for oo in O.index:
            F = R_F(R,年利率与客户流失率,O,oo)
            v = bad_budget(O.loc[oo,'信誉评级预测'],O.loc[oo,' 企业类别'])
            s = f(v,F,年利率与客户流失率.loc[R,'贷款年利率'],M)
            I = I + s
            利润['v:'+str(v)+'><F:'+str(F)+'><R:'+str(R)+'><M:'+str(M)]=I
```

附录 F　预测信誉评级 2

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
```

```
待预测_归一化 = pd.read_csv('./task2/待预测_归一化.csv',encoding='gbk').set_index(' 企业代号')
待预测_归一化

作废率_频次 = pd.read_csv('./task2/作废率_频次.csv',encoding='gbk').set_index('企业代号')
作废率_频次

待预测_归一化 = pd.concat([待预测_归一化,作废率_频次],axis=1)
待预测_归一化

X = 待预测_归一化[['进项_价税合计_归一化','进项_价税合计_标准化','销项_价税合计_标准化'
,'销项_价税合计_归一化','上游','下游','稳定_标准差','作废率','资金流通频次']]
model = pickle.load(open("./task1/model_randomForest.pkl", "rb"))
y = model.predict(X)
y

待预测_归一化['预测信誉类别'] = y
待预测_归一化

预测后 = 待预测_归一化[['预测信誉类别']]
预测后

def tran(x):
    if x == 0:
        return 'A'
    elif x == 1:
        return 'B'
    elif x == 2:
        return 'C'
    else:
        return 'D'

预测后['预测信誉类别'] = 预测后['预测信誉类别'].apply(tran)
预测后

预测后.to_csv('./task2/预测后.csv',encoding='gbk')

待预测_归一化
```

附录 G　问题二决策树与逻辑回归模型建立

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']
```

```
plt.rcParams['axes.unicode_minus']=False
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler

企业信息2 = pd.read_csv('./task2/企业信息2.csv',encoding='gbk',index_col=0).set_index('企业代号')
实力无标准化 = pd.read_csv('./task2/实力无标准化.csv',encoding='gbk',index_col=0)
作废率_频次 =
    pd.read_csv('./task2/作废率_频次.csv',encoding='gbk').set_index('企业代号').drop(columns='企业名称')
平均年利润 =
    pd.read_csv('./task2/平均年利润.csv',encoding='gbk',index_col=0).drop(columns='企业名称')
上下游无归一化 = pd.read_csv('./task2/上下游无归一化.csv',encoding='gbk',index_col=0)
预测后 = pd.read_csv('./task2/预测后.csv',encoding='gbk',index_col=0)
data1 = pd.concat([实力无标准化,作废率_频次,平均年利润,上下游无归一化,企业信息2,预测后],axis=1)
data1.to_csv('./task2/features_labels.csv',encoding='gbk')
data1

data1['行业类型'].unique()

data1['企业类别'].unique()

def tran1(x):
    if x == '其他':
        return 0
    elif x == '服务业':
        return 1
    elif x == '建筑业':
        return 2
    elif x == '制造业':
        return 3
    elif x == '交通运输业':
        return 4
    elif x == '信息技术':
        return 5
    elif x == '餐饮业':
        return 6
def tran2(x):
    if x == '个体经营':
        return 0
    elif x == '有限公司':
        return 1
    elif x == '有限责任公司':
        return 2
    elif x == '其他':
        return 3
data1['行业类型'] = data1['行业类型'].apply(tran1)
```

```
data1['企业类别'] = data1['企业类别'].apply(tran2)
data1

X = data1.drop(columns=['预测信誉类别','企业名称'])
y = data1['预测信誉类别']

train_X, val_X, train_y, val_y = train_test_split(X.to_numpy(), y, test_size=0.3,
    random_state=0)
dt = DecisionTreeClassifier()
dt.fit(train_X, train_y)
dt.score(val_X, val_y)
cross_val_score(dt, X, y, cv=15)

dt.feature_importances_
def sigmoid(x):
    w = np.array([0.14867777, 0.12895121,0.29068381, 0.1404667 , 0.11771 , 0.1141494
        ,0.03118526, 0.02817585])
    z = np.dot(w,x)
    return 1/(1+np.exp(-z))


# ## 使用标准化后的数据

ss = StandardScaler()
X = ss.fit_transform(X)
X

xx=[]
yy=[]
for i in X:
    xx.append(i)
    yy.append(sigmoid(i))
plt.plot(xx,yy)

yy

data1['风险概率'] = yy
data1

data1.to_csv('./task2/features_labels_风险概率.csv',encoding='gbk')
```

附录 H　预测违约概率

```
import pandas as pd
```

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score

实力 = pd.read_csv('./task1/实力.csv',encoding='gbk').set_index('企业代号')
实力

上游 = pd.read_csv('./task1/上游.csv',encoding='gbk').set_index('企业代号')
下游 = pd.read_csv('./task1/下游.csv',encoding='gbk').set_index('企业代号')
上下游 = pd.concat([上游,下游],axis=1)
上下游 = 上下游.rename(columns={'销方单位代号':'上游','购方单位代号':'下游'})
上下游['上游']=上下游['上游'].apply(lambda
    x:(x-上下游.describe().loc['min','上游'])/(上下游.describe().loc['max','上游']
-上下游.describe().loc['min','上游']))
上下游['下游']=上下游['下游'].apply(lambda
    x:(x-上下游.describe().loc['min','下游'])/(上下游.describe().loc['max','下游']
-上下游.describe().loc['min','下游']))

上下游

企业信息1 = pd.read_excel(r'附件1: 123家有信贷记录企业的相关数据.xlsx',sheet_name=0)
企业信息1 = 企业信息1.set_index('企业代号')
企业信息1

df = np.load('./task1/all_plot.npy',allow_pickle=True)
dff = []
for i in df:
   dff.append(i.describe().loc['std'].mean())
dff

企业信息1['稳定_标准差'] = dff
企业信息1 = 企业信息1.fillna(0)
企业信息1

def tran(x):
   if x=='是':
      return 1
   else:
      return 0

企业信息1['是否违约'] = 企业信息1['是否违约'].apply(tran)
A = 企业信息1[企业信息1['信誉评级']=='A']['是否违约'].sum()/企业信息1[企业信息1['信誉评级']=='A']
['是否违约'].count()
B = 企业信息1[企业信息1['信誉评级']=='B']['是否违约'].sum()/企业信息1[企业信息1['信誉评级']=='B']
['是否违约'].count()
C = 企业信息1[企业信息1['信誉评级']=='C']['是否违约'].sum()/企业信息1[企业信息1['信誉评级']=='C']
['是否违约'].count()
```

```
D = 企业信息1[企业信息1['信誉评级']=='D']['是否违约'].sum()/企业信息1[企业信息1['信誉评级']=='D']
['是否违约'].count()

def tranf(x):
    if x=='A':
        return 0
    elif x=='B':
        return 1
    elif x=='C':
        return 2
    elif x=='D':
        return 3
企业信息1['违约率'] = 企业信息1['信誉评级'].apply(tranf)
企业信息1

企业信息1['稳定_标准差'] = 企业信息1['稳定_标准差'].apply(lambda
    x:(x-企业信息1['稳定_标准差'].describe().loc['mean'])/企业信息1['稳定_标准差'].describe().loc['std'])
企业信息1

data1 = pd.concat([实力,上下游,企业信息1],axis=1)
data1

X = data1[['进项_价税合计_归一化','销项_价税合计_归一化','上游','下游','稳定_标准差']].to_numpy()
y = data1['违约率']
display(X,y)

train_X, val_X, train_y, val_y = train_test_split(X, y, test_size=0.3, random_state=0)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=4, weights='distance')
knn.fit(X, y)
acc1 = knn.score(val_X, val_y)
print(acc1)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=1e9,multi_class='ovr',warm_start=True)
lr.fit(X, y)
acc2 = lr.score(val_X, val_y)
acc2

from collections import Counter
Counter(企业信息1['信誉评级'])
```

附录 I　训练预测信誉类别 3

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split, cross_val_score

data1归一化=pd.read_csv('./task1/data1归一化.csv',encoding='gbk',index_col=0)
data1无归一化 = pd.read_csv('./task1/features_labels.csv',encoding='gbk',index_col=0)
data1归一化['作废率']=data1无归一化['作废率']
data1归一化['资金流通频次']=data1无归一化['资金流通频次']
data1归一化

m = data1归一化.reset_index()
four_ = pd.concat([m,m,m])

X = four_[['进项_价税合计_归一化','进项_价税合计_标准化','销项_
    价税合计_标准化','销项_价税合计_归一化','上游','下游','稳定_标准差','作废率','资金流通频次']]
y = four_[['信誉评级']]
display(X,y)

from imblearn.over_sampling import SMOTE
from collections import Counter
smo = SMOTE(random_state=42)
X_smo, y_smo = smo.fit_sample(X,y)

len(X_smo)

from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit(four_[['信誉评级']])
y = enc.transform(four_[['信誉评级']]).toarray()
print(y)

models = [
    RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0),
    LinearSVC(),
    LogisticRegression(random_state=0),
]
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))
entries = []
for model in models:
    model_name = model.__class__.__name__
```

```
    accuracies = cross_val_score(model, X_smo, y_smo, scoring='accuracy', cv=CV)
    for fold_idx, accuracy in enumerate(accuracies):
        entries.append((model_name, fold_idx, accuracy))
cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])

import seaborn as sns

sns.boxplot(x='model_name', y='accuracy', data=cv_df)
sns.stripplot(x='model_name', y='accuracy', data=cv_df,
              size=8, jitter=True, edgecolor="gray", linewidth=2)
plt.savefig('./task1/预测4类别模型对比.png',dpi=300)
plt.show()

from sklearn.model_selection import cross_val_score
scorel = []
for i in range(0,75,10):
    rfc = RandomForestClassifier(n_estimators=i+1,
                                 n_jobs=-1,
                                 random_state=90)
    score = cross_val_score(rfc,X_smo,y_smo,cv=10).mean()
    scorel.append(score)
print(max(scorel),(scorel.index(max(scorel))*10)+1)
plt.figure(figsize=[20,5])

plt.plot(range(1,76,10),scorel)
plt.xlabel('n_estimators')
plt.ylabel('score')
plt.title('RandomForest')
plt.savefig("随机森林调参.png",dpi=300)
plt.show()

cv_df.groupby('model_name').accuracy.mean()

from sklearn.metrics import confusion_matrix as CM
train_X, val_X, train_y, val_y = train_test_split(X_smo, y_smo, test_size=0.3, random_state=0)
clf = RandomForestClassifier(n_estimators=71)
clf.fit(train_X,train_y)
y_pred=clf.predict(val_X)
CM(val_y,y_pred)

clf = RandomForestClassifier(n_estimators=71)
clf.fit(train_X,train_y)
pickle.dump(clf, open("./task1/model_randomForest.pkl", "wb"))
```

附录 J　计算坏账率

```
import pandas as pd
企业信息1 = pd.read_csv('./task2/企业信息1.csv',encoding='gbk',index_col=0)
企业信息2 = pd.read_csv('./task2/企业信息2.csv',encoding='gbk',index_col=0)
企业信息1

def tran(x):
    if x == '是':
        return 1
    else:
        return 0
企业信息1['是否违约'] = 企业信息1['是否违约'].apply(tran)
le = ['A', 'B','C']
ca = ['个体经营','有限公司','有限责任公司','其他']

dic = {}
for l in le:
    for c in ca:
        dic[l+':&:'+c] = 企业信息1[企业信息1['信誉评级']==l]
[企业信息1['企业类型']==c]['是否违约'].sum()/企业信息1[企业信息1['信誉评级']==l]
[企业信息1['企业类型']==c]['是否违约'].shape[0]



dic
```

附录 K　提取资金流通频率和无效发票比例

```
import pandas as pd
进项发票1 = pd.read_excel(r'附件1: 123家有信贷记录企业的相关数据.xlsx',sheet_name=1)
销项发票1 = pd.read_excel(r'附件1: 123家有信贷记录企业的相关数据.xlsx',sheet_name=2)
企业信息1 = pd.read_excel(r'附件1: 123家有信贷记录企业的相关数据.xlsx',sheet_name=0)

from collections import Counter
作废 = []
for M in 企业信息1['企业代号'].values:
    fa = dict(Counter(进项发票1[进项发票1['企业代号']==M]['发票状态']))
    try:
        作废.append(fa['作废发票']/(fa['有效发票']+fa['作废发票']))
    except:
        作废.append(0)
作废

作废2 = []
```

```
for M in 企业信息1['企业代号'].values:
    fa = dict(Counter(销项发票1[销项发票1['企业代号']==M]['发票状态']))
    try:
        作废2.append(fa['作废发票']/(fa['有效发票']+fa['作废发票']))
    except:
        作废2.append(0)
作废2

平均废例 = [(i+j)/2 for i,j in zip(作废,作废2)]
平均废例

进项发票1

作废率_频次 = 企业信息1[['企业代号','企业名称']].set_index('企业代号')
作废率_频次['作废率'] = 平均废例
作废率_频次

频次 = list(dict(Counter(销项发票1['企业代号'])).values())
作废率_频次['资金流通频次'] = 频次
作废率_频次

作废率_频次.to_csv('./task1/作废率_频次.csv',encoding='gbk')



# ## task2

import pandas as pd
进项发票2 = pd.read_excel(r'附件2: 302家无信贷记录企业的相关数据.xlsx',sheet_name=2)
销项发票2 = pd.read_excel(r'附件2: 302家无信贷记录企业的相关数据.xlsx',sheet_name=1)
企业信息2 = pd.read_excel(r'附件2: 302家无信贷记录企业的相关数据.xlsx',sheet_name=0)

from collections import Counter
作废 = []
for M in 企业信息2['企业代号'].values:
    fa = dict(Counter(进项发票2[进项发票2['企业代号']==M]['发票状态']))
    try:
        作废.append(fa['作废发票']/(fa['有效发票']+fa['作废发票']))
    except:
        作废.append(0)

作废2 = []
for M in 企业信息2['企业代号'].values:
    fa = dict(Counter(销项发票2[销项发票2['企业代号']==M]['发票状态']))
    try:
        作废2.append(fa['作废发票']/(fa['有效发票']+fa['作废发票']))
    except:
        作废2.append(0)
```

```
平均废例 = [(i+j)/2 for i,j in zip(作废,作废2)]
平均废例



作废率_频次 = 企业信息2[['企业代号','企业名称']].set_index('企业代号')
作废率_频次['作废率'] = 平均废例
作废率_频次

频次 = list(dict(Counter(销项发票2['企业代号'])).values())
作废率_频次['资金流通频次'] = 频次
作废率_频次

作废率_频次.to_csv('./task2/作废率_频次.csv',encoding='gbk')
```

附录 L　滑动平均

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus']=False
df = np.load('./task2/all_plot.npy',allow_pickle=True)

series =
    df[147].iloc[:,0].append(df[147].iloc[:,1]).append(df[147].iloc[:,2]).append(df[147].iloc[:,3])
S = series.reset_index()
S

S['月份'] = pd.date_range('2017-01-01','2021-01-01',freq='M')
S = S.loc[:37,:]

m=0
xx = []
ss = []
for i in S.index:
    m = 0.9*m+0.1*S.iloc[i,1]
    m_ = m/(1-0.9**i)
    xx.append(S.iloc[i,2])
    ss.append(m_)
ss

plt.plot(xx[:35],d)
plt.xlabel('time')
plt.xticks(rotation=90)
```

```
plt.ylabel('m_')
plt.title('E269')
plt.savefig('./滑动平均.png',dpi=300)

delta = [ss[i]-ss[i-1] for i in range(2,len(ss))]
d = [sum(delta[0:i])/i for i in range(1,len(delta))]
d

fig = plt.figure()
plt.plot(xx[3:],d)
plt.xlabel('time')
plt.yticks(np.arange(0,1))
plt.xticks(rotation=30)
plt.ylabel('€')
plt.title('E269')
plt.savefig('./滑动平均.png',dpi=300)
a
```

附录 M　自然语言处理

```
import pandas as pd
import re
import jieba

data1 = pd.read_excel(r'附件1: 123家有信贷记录企业的相关数据.xlsx',sheet_name=0)
data2 = pd.read_excel(r'附件2: 302家无信贷记录企业的相关数据.xlsx',sheet_name=0)

company_name1 = data1["企业名称"].apply(lambda x: re.sub("[^\u4e00-\u9fa5]","",x))
    #正则化，仅保留中文字符
company_name2 = data2["企业名称"].apply(lambda x: re.sub("[^\u4e00-\u9fa5]","",x))

jieba.add_word("有限责任公司")
company_name1 = company_name1.apply(lambda x:jieba.lcut(x))
company_name2 = company_name2.apply(lambda x:jieba.lcut(x))

company_type = pd.read_csv("company_type.txt",header=None)
company_type = list(company_type.iloc[:,0])
company_name1 = company_name1.apply(lambda x: [i for i in x if i in company_type])
company_name2 = company_name2.apply(lambda x: [i for i in x if i in company_type])

for i in range(122):
    if company_name1[i] == []:
        company_name1[i] = ["其他"]

for i in range(302):
```

```
    if company_name2[i] == []:
        company_name2[i] = ["其他"]
```