基于企业信誉评级的信贷策略建模研究

摘要

中小企业具有规模较小、抵抗风险能力较差，同时缺少资产抵押等特点，商业银行对其信贷业务的风险把控的难度较高。对此，本文建立数学模型量化中小企业的信贷风险，并设计在银行年贷总金额指定时各类型企业的借贷方案以及在突发因素影响下各企业借贷方案的调整策略。本文工作如下：

分别考虑企业的发展前景和企业的历史信用，建立企业信贷风险量化模型。其中，企业的发展前景主要考虑该企业所处行业的发展时期和该行业的竞争程度，对于行业发展时期，由中证行业指数确定，而对于该行业的竞争程度，则由行业集中度确定。企业历史信用则由信誉评级和历史违约记录确定。基于上述指标获得企业的综合风险打分。对于企业借贷方案的设计，本文考虑借贷年利率与用户流失率的关系以及企业信誉打分与企业违约率的关系，并基于线性规划设计不同信誉等级下企业的信贷方案（贷款金额和贷款利率），以期最大化银行收益，同时最小化客户违约风险。

基于企业进项和销项发票数据设计特征工程，分为销售类和支出类两大特征体系共 9 种特征，包括月开票总额单月同比、月进销匹配度等。基于上述特征训练决策树模型以及集成学习典型模型：随机森林和 XGBoost 模型实现企业信誉评级的自动识别。实验结果显示 XGBoost 在三种模型中识别准确率最高，其在训练集中 F1 值为 1，而在测试集中 F1 值为 0.73，存在一定过拟合风险。基于企业的信誉评级识别结果，结合问题一设计模型，确定不同信誉等级的企业借贷方案。其中，对于信誉为 D 的企业不考虑给予借贷，而对于信誉为 A 的企业其借贷金额参考值为 58.27（万元），借贷年率为 5.2%。具体根据企业需求和实际进行管理情况进行有限的上下调整。而信誉为 B、C 的企业则在此基础上借贷金额进行递减，借贷年利率上调。

考虑不同企业对突发因素的抵抗能力不同，提出企业的“可靠性”指标。同时基于企业规模（进销发票总额之差）和企业信誉等级对企业“可靠性”指标进行量化。并假设企业“可靠性”越高，其在面对突发因素时的违约率将越低。基于此，建立企业“可靠性”与违约率的函数关系，对问题一所得模型进行修正。最后，考虑到实际情况的复杂多变和不确定性，企业的违约率将服从 Gaussian 分布而非固定值。同时，在突发因素影响下为保证银行盈利，尽可能的帮助企业渡过难关，对此，本文提出增加“优质”企业贷款额度，减少或取消“劣质”企业贷款机会，并降低贷款利率的思路，以期实现银行获利的最大化、企业违约风险的最小化，以及企业纾困的最大化。基于此，对突发因素影响下企业的借贷策略进行多次模拟，最终获得不同信誉等级、不同规模企业的借贷方案。其中信誉等级为 A、企业规模为 5（最高）下企业的贷款额度为 73.23（万元），而借贷年利率为 4.2%。同样，对于信誉等级为 D 的企业剥夺其借贷机会。

关键词：借贷风险量化　信誉评级自动识别　突发因素

1. 问题重述

对于小微企业的借贷业务，由于该类型企业缺少有效的固定资产抵押同时经营规模较小，因此银行一般通过分析其历史信誉情况和经营状况，完成企业信贷风险评估，设计借贷策略。

（1）问题一要求利用企业基本信息和进销发票信息完成企业的风险量化分析，同时设计在固定年贷款总额的条件下，各企业的借贷策略；

（2）问题二要求基于问题一的研究基础，给出在缺少企业信誉评级信息，同时指定年贷总额的前提下，各企业的借贷策略；

（3）问题三要求基于上述两项研究结论，给出在外部突发因素冲击的条件下，不同行业、不同类型企业的借贷方案调整策略。

2. 模型假设

（1）假设行业所处发展时期越兴盛同时该行业内部竞争越小，该公司的信贷风险将越低。反之，若该行业所处发展时期越衰败，同时该行业内部竞争越激烈，则认为该企业信贷风险越高。

（2）假设行业年收益率超过 10%为兴盛期，低于-10%为衰败期，介于两者之间为稳定期；

（3）假设企业信誉越差其违约的可能性将越高；

（4）假设突发因素将影响企业的违约率 f_r，同时规模越小、信誉等级越低的企业其违约率将越高，反之，规模越大、信誉等级越高的企业其违约率将越低；

3. 符号及变量说明

符号	含义
CR_n	规模最大的 top n 家企业的行业集中度
μ	企业近一年收益率
S	企业信贷风险综合得分
P	行业前景得分
R	企业信用得分
C	贷款总金额（万元）
r	信用等级 $(r=1,2,3,4,5)$
m_r	可获得的贷款额度期望（万元）
f_r	表示风险等级为 r 的企业违约率
λ_r	贷款年利率
t_r	贷款时间（年）
Pro	银行收益
d	用户流失率收益折扣因子
l_i	信用等级为 i 时的客户流失率
M_i	i 企业规模得分
ϖ	企业近三月开票总额单月同比

4. 问题分析

4.1 问题一分析

问题一要求对附件一所给123家企业进行信贷风险量化分析同时给出在银行年度信贷总额固定时的各企业信贷方案。对此本文首先基于附近一所给信息对考虑企业的行业前景和企业信用这两大指标对企业信贷风险进行建模。具体来说对于行业前景，主要考虑基于中证行业指数的行业所处发展时期和衡量企业竞争激烈程度的行业的集中度，而对于企业信用则主要考虑企业的信誉评级和企业违约记录。对于信贷策略的建模，本文主要考虑银行和企业的博弈，即在最大化银行收益的同时最小化客户流失。因此，首先基于函数拟合建立贷款利率和用户流失率间的函数关系，对企业收益进行修正。同时，本文还假设用户信誉等级越低，违约风险将越高，考虑建立信誉等级和违约风险相关模型，对企业收益进行修正。最后，综合考虑企业信誉-违约和贷款年利率-流失这两种因素建立不同信誉等级企业的借贷放啊。

4.2 问题二分析

问题二要求在问题一的基础上给出在银行信贷总额确定时企业的信贷策略。分析附件二相关数据知，附件二所给企业缺少信誉评级信息。对此，本文首先设计特征工程，利用企业进销发票数据设计营销和支出类两大特征，共9小特征（包括月开票总额单月同比、近三月开票总额单月同比等）。同时企业基于集成学习中的决策树基学习器、随机森林和 XGBoost 模型对附件二中的企业信誉评级进行分类，然后基于问题一的相关研究成果给出不同信誉等级的企业信贷方案。

4.3 问题三分析

问题三要求分析在突发情况下，企业正常经营活动受到影响时的信贷风险调整策略。对于不同企业本文提出企业抵抗风险的能力即企业“可靠性”，该指标由企业的规模确定，如企业规模越大则相对“可靠性”将越高。同时考虑建立企业“可靠性”与企业违约率间的关系，对问题一相关模型进行修正。最后考虑到实际情况中的不确定性，本文设计企业违约率为符合 Gaussian 分布的变量，然后进行模拟仿真，确定不同信用等级、不同经验规模下各企业信贷方案的调整策略。

5. 模型的建立与求解

5.1 中小微企业信贷风险量化分析及信贷策略建模

中小微企业信贷风险量化分析对于商业银行信贷风险的管理具有重要的研究价值。本小问首先对附表一中的相关数据包括企业名称、信誉评级和是否违约建立信贷风险量化模型，判断银行是否发放贷款。同时结合附件三所给信息考虑贷款利率对客户流失率的影响，制定在年度信贷总额固定时其对微小企业的信贷策略。

5.1.1 结合行业环境及信誉评级的信贷风险量化建模

信贷风险量化管理与信贷策略的指定是商业银行的核心业务之一，其在银行的经营和管理中占据重要地位。一般而言，银行主要基于贷款企业的历史交易信息、经营状况、

违约信息、同时结合环境风险、信用风险、市场风险、行业和法律等风险确定是否向该企业发放贷款，以及贷款策略，如贷款金额、贷款期限及年利率等。本文首先考虑基于公司名称确定公司所处行业，同时结合该行业近期相关的发展状况和前景，该公司的信誉评级以及违约状况等数据建立信贷风险量化模型。

（1）行业发展前景分析

由附表 1 知，其企业的名称在一定程度上体现了该公司的所处行业类别，因此本文可以考虑该行业的发展前景在一定程度上判断该行业的信贷风险，具体包括[1]：

① 行业所处时期：处于兴起、发展、成熟或是衰退时期（兴、稳、衰）；

② 行业内部竞争程度：及行业的细分程度（良、一般、差）；

其中，行业所处时期和竞争程度分别对应的风险等级如下表所示：

表 5.1 行业发展前景对应风险打分表

行业所处发展时期			行业内部竞争程度		
兴	稳	衰	良	一般	差
风险等级			风险等级		
低	中	高	低	中	高
打分			打分		
1	3	5	1	3	5

本文假设行业所处发展时期越兴盛同时该行业内部竞争越小，该公司的信贷风险将越低。反之，若该行业所处发展时期越衰败，同时该行业内部竞争越激烈，则认为该企业信贷风险越高。如表 5.1 所示，其中行业所处发展时期和行业内部竞争程度可参考我国相关行业指数，如中证行业指数[2]等，其主要反映该行业的热度和发展前景。本文选择中证行业指数同期近 1 年收益率作为该行业所处发展时期的指标，同时定义：

表 5.2 中证行业指数年收益率对应行业发展时期表

近 1 年收益率 μ（%）	μ>+10%	-10%< μ <+10%	μ <-10%
所处发展时期	兴	稳	衰

行业内部竞争程度选择行业集中度指数[3]，其一般是由某行业排名 top 4 的企业销售额(或生产量等指标)占全行业总销售额或其它指标的比例来含量。若 CR4 越大，即说明该行业的集中度越高，市场竞争越趋向于垄断；反之，则集中度越低，市场竞争越趋向于竞争。集中度是衡量行业市场内部竞争程度的一个重要指标，其计算公式如下：

$$CR_n = \frac{\sum (X_i)_n}{\sum (X_i)_N}, N > n \qquad (1)$$

式（1）中，

CR_n：表示规模最大的前几家企业的行业集中度，若 $n=4$ 则为前 4 家企业的行业集中度 CR4；

X_i：表示第 i 家企业的产量、销售额、产值、销售量、雇佣人数、资产总额等指标值；

n：表示选择的本行业市场规模最大的前n家公司；

N：表示此行业内的企业总数；

一般情况下，n一般取值为 4 或 8，即计算行业中前 4 家或 8 家公司的行业集中度。基于日本通产省和美国经济学家贝恩对产业集中度的划分标准[4]，其将行业内部竞争激烈程度粗分为寡占型（CR8 ≥ 40%）和竞争型（CR8＜40%）两大类。其中，寡占型又可细分为极高寡占型（CR8 ≥ 70%）和低集中寡占型（40% ≤ CR8＜70%）；而竞争型又可细分为低集中竞争型（20% ≤ CR8＜40%）和分散竞争型（CR8＜20%），如表 5.3 所示：

表 5.3 CR4/CR8 产业集中度与竞争程度

产业集中度	CR4（%）	CR8（%）	竞争程度
寡占 1 型	CR4≥85	—	差
寡占 2 型	75≤CR4<85	CR8≥85	一般
寡占 3 型	50≤CR4<75	75≤CR8<85	一般
寡占 4 型	35≤CR4<50	45≤CR8<75	一般
寡占 5 型	30≤CR4<35	40≤CR8<45	良
竞争型	CR4<30	CR8<40	良

本文定义寡占 1 型的行业内部竞争激烈程度差，而竞争型和寡占 5 型行业的内部竞争程度良好，其它为一般。我国各行业的产业集中程度同时可参考国泰君安 2019 年发布相关报告[5]分析我国 12 大主题包括、白酒、大众食品、可选消费品、消费属性医药、高端服务、消费电子、电子装备、软件、工业服务与产品等行业的行业集中度。

本文首先对附表一中的各行业名称进行一定的处理，剔除“*”符号、有限责任公司、有限公司，以及词频小于 2 的低频词，绘制云词图分析其行业分布，如下图所示：

图 5.1 企业名称云词图

如图 5.1 所示，可以看出在附件一所给的 120 余家公司中科技、建筑等公司等占大多数，而咨询服务类公司数目较少。本文参考中华人民共和国国家标准 GB/T 4754-2017《国民经济行业分类标准》对上述公司进行简单分类。该标准共包含 97 大类，包括农、林、牧、渔业，采矿业，制造业等，数百种中类，数千种小类。如制造业<农副产品加工业<小麦加工等。

代码				类别名称	说明
门类	大类	中类	小类		
A				**农、林、牧、渔业**	本门类包括 01～05 大类
	01			**农业**	指对各种农作物的种植
		011		谷物种植	指以收获籽实为主的农作物的种植，包括稻谷、小麦、玉米等农作物的种植和作为饲料和工业原料的谷物的种植
			0111	稻谷种植	
			0112	小麦种植	
			0113	玉米种植	
			0119	其他谷物种植	
		012		豆类、油料和薯类种植	
			0121	豆类种植	
			0122	油料种植	
			0123	薯类种植	
		013		棉、麻、糖、烟草种植	
			0131	棉花种植	

图 5.2 GB/T 4754-2017 国民经济行业分类标准

如图 5.2 所示基于 GB 4754 对附件一种所给企业信息映射统一至行业大类中，如建筑、建设工程、建筑工程等同一为建筑业。经处理后各行业的企业数目如下图所示：

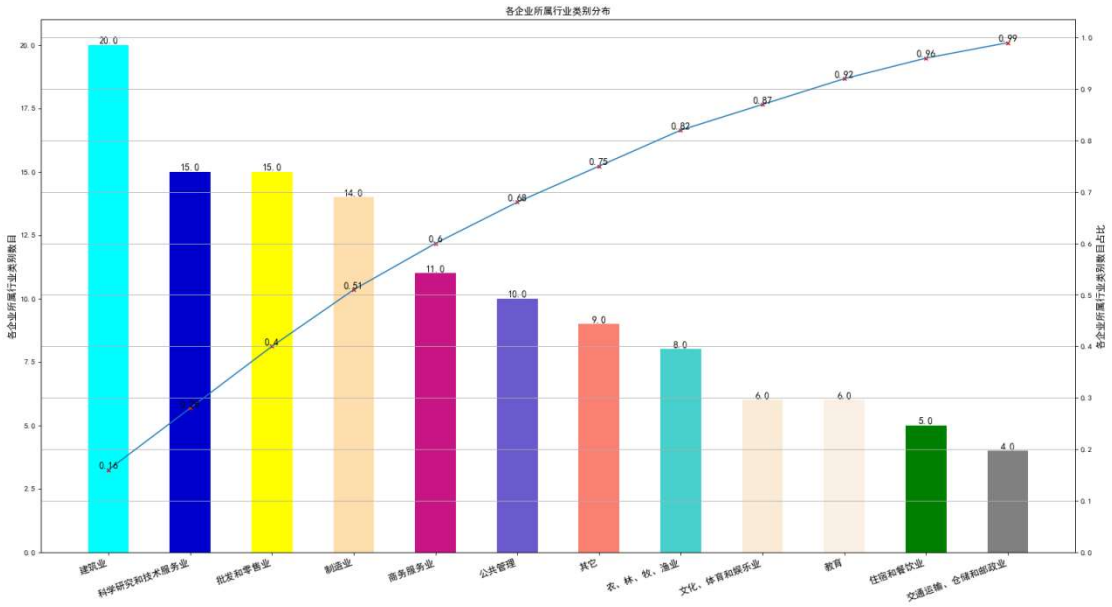

图 5.3 各企业所属行业分布

如图 5.3 所示，可以看出上述 123 家企业中属于建筑行业的数目最多，而属于交通

运输、仓储和邮政业的最少。这里分别对上述 123 家企业分析其行业发展前景如下表所示：

表 5.4 各企业行业竞争发展前景分析（截取部分）

企业代号	企业名称	所属行业	中证行业指数（年收益率）	发展前景	CR4 行业集中度	竞争程度
E71	***农业科技有限公司	农、林、牧、渔业	-12.11%	衰	42%	良
E1	***电器销售有限公司	批发和零售也	-8.15	稳	63%	一般
E99	***建筑工程有限责任公司	建筑业	-4.27	稳	88	差

如上表所示，对附件一中各企业的发展前景进行分析，以此作为信贷风险的参考指标之一。

（2）信誉评级和是否违约统计分析

对附表一中各企业的信誉评级和是否违约这两项指标进行简单的统计分析，如图 5.4 所示：

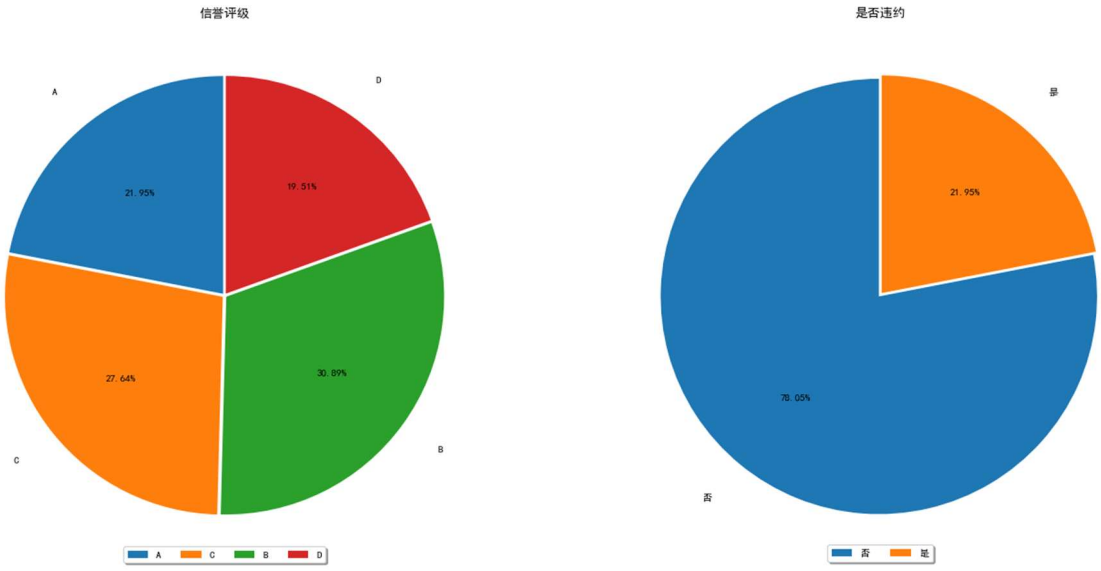

图 5.4 各企业信誉评级和是否违约数据分布

如图 5.4 所示，附件一种所给 123 家公司中其信誉等级（A、B、C、D）分布较为均匀，而是否违约记录中仅 21.95 的公司存在违约记录。

（3）考虑行业环境和企业信誉的风险量化模型

本文考虑利用各厂商行业环境信息、信誉等级和是否违约记录这三项指标综合打分给出信贷风险等级量化值，如下表所示：

表 5.5 信贷风险等级评分表

发展时期	得分	竞争程度	得分	信誉等级	得分	是否违约	得分	综合得分	信贷风险等级
兴	5	良	5	A	5	0	5	0-1	5
稳	3	一般	3	B	4	1	0	1-2	4
衰	1	差	1	C	3			2-3	3
				D	1			3-4	2
5								4-5	1

如表 5.5 所示本文首先基于企业名称确定企业所属行业类别，然后定义行业发展时期和行业发展指标反映该企业所属行业发展前景。基于中证行业指数年收益率和 CR4 行业集中度对行业发展前景进行打分（分为 3 级）。同时，考虑企业信誉等级和历史是否违约记录对企业信誉进行打分。最终对上述打分加权求和获得企业综合得分，依此来评估企业信贷风险等级。其中若综合得分越低则表明该企业信贷风险等级越高，风险越大。各企业信贷风险综合得分指标体系如图 5.5 所示：

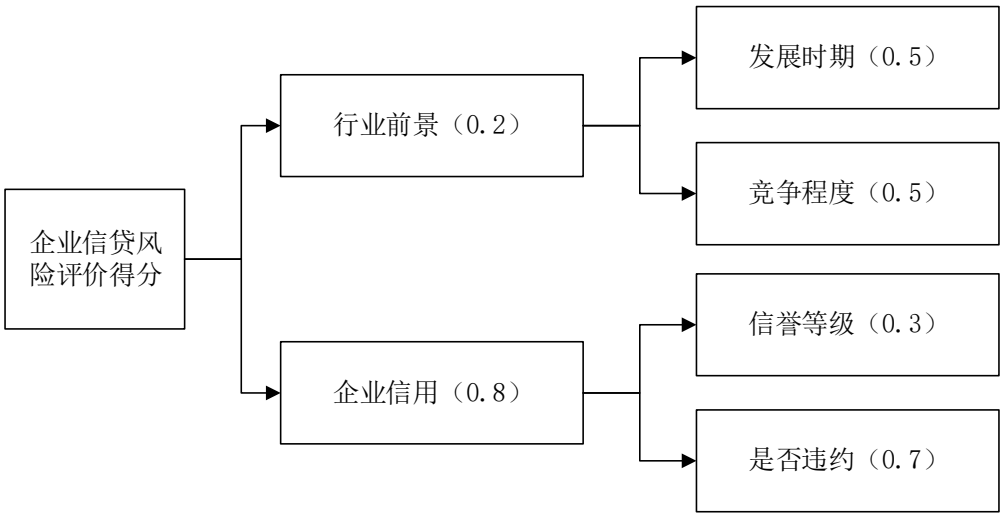

图 5.5 企业信贷风险综合得分指标体系

如图 5.5 所示，其中括弧内部为该指标的权值。各银行可根据其自身的经营策略更改权值取值及综合得分与风险等级对应表，若为“冒进型”策略则可以降低信贷风险等级，若为“稳健性”则可提高信贷风险等级。其综合打分函数如下所示：

$$\begin{cases} S = w \bullet P + (1-w) \bullet R \\ P = w_p \bullet s + (1-w_p) \bullet c \\ R = w_r \bullet r + (1-w_r) \bullet b \end{cases} \quad (2)$$

上式中，S,P,R 分别表示综合得分，行业前景得分和企业信用得分；

s,c 分别表示发展时期得分和竞争程度得分；

r,b 分别表示信誉等级得分和是否违约得分；

w,w_p,w_r 分别表示综合打分权值，行业前景打分权值和企业信用打分权值；

基于式（2）即可获得量化后的各企业的信誉风险等级。

5.1.2 考虑客户违约及流失的企业信贷策略建模研究

对于商业银行如何设计借贷方案在保证客户满意且能按时还款的前提下使自身收益最大化，同时风险最低这将是一个优化问题。对此本论文首先考虑基于企业信贷风险量化分析结果优化不同风险等级客户的信贷额度和年率使其收益最大化。在此基础上还将同时考虑年利率对客户流失的影响，以此修正上述模型，使得结果更优。

（1）考虑不同信用得分的客户违约率

记贷款总金额为C（万元），信用等级为$r(r=1,2,3,4,5)$的客户可获得的贷款额度期望为m_r（万元），信用得分为f_r，年利率为λ_r，贷款时间（年）为t_r。同时假设信誉等级与违约的概率（未按时还款）服从如下函数规律：

$$\begin{cases} f_1(x)=y=-\dfrac{x}{5}+1 \\ f_2(x)=y=\cos(\dfrac{\pi}{10}x) \\ f_3(x)=y=\dfrac{1}{x+1} \\ f_4(x)=y=\dfrac{1}{1+e^{ax-b}} \end{cases},x\in[0,5] \tag{2}$$

上述四种函数图像如下所示：

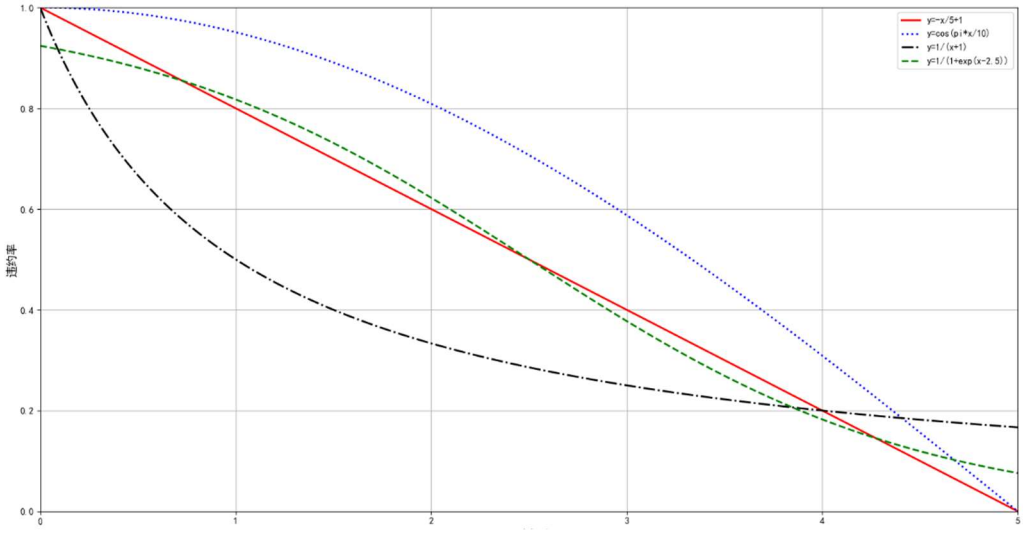

图 5.6 不同信用得分与违约率映射函数

如图 5.6 可以看出，这里使用不同函数类型将信用得分映射至违约率上，银行可以根据不同策略调整函数的选择。若选择“稳健策略”则可以考虑使用函数$f_1(x)$，若考虑“激进冒险策略”则可以选择$f_3(x)$函数，若要求“保守”则可以选择$f_2(x)$函数。同时若要求策略灵活多样则可以选择函数$f_4(x)$，同时调整超参数a,b，这里a,b取值分别为 1 和-2.5。

同时，本文要求风险最高得分对应的违约风险越大，即信用得分为 0 时其违约率为

0.8。而信用得分为 5 时违约率为 0.1，因此这里对上述违约率进行标准化处理，如下式所示：

$$f_{st}(x)=0.7f(x)+0.1 \tag{3}$$

上式中，$f_{st}(x)$即为标准化后的信用得分-违约率函数。

记银行的收益为Pro，则其计算公式如下所示：

$$\begin{aligned}
&\max \quad \mathrm{Pro}=\sum_{r=1}^{5} n_r \bullet f_r \bullet (1+\lambda_r m_r)^{t_r} \\
&s.t. \quad \sum_{r=1}^{5} n_r m_r = C \\
&\quad m_r \in [m_d, m_u] \\
&\quad \lambda_r \in [\lambda_d, \lambda_u] \\
&\quad t_r \in [t_d, t_u] \\
&\quad r = 1,2,3,4,5
\end{aligned} \tag{4}$$

上式中，Pro 即为银行获利；

n_r表示风险等级为r的企业数目；

f_r表示风险等级为r的企业违约率；

由题目知，银行贷款时间为 1 年，则$t_r=1$；贷款额度为10~100万元，则$m_r \in [10,100]$；年利率为 4%~15%，则$t_r \in [0.04,0.15]$。通过优化贷款策略即可获得最大的收益 Pro。设计对于不同信用得分的用户考虑其贷款时间、贷款额度和贷款利率满足如下规律[7]：

表 5.6 信贷策略与信用得分参考表

信用得分	贷款时间	贷款额度	贷款利率
高→低	长→短	多→少	高→低

如表 5.6 所示，根据不同等级的信用得分调整信贷策略，以确定m_r, t_r的合理取值区间，使得银行收益最大。

（2）考虑顾客流失的收益率设计

由附件 3 知，不同的贷款年利率下不同信用等级的客户流失率将不断发生变化，而客户的高流失率也将带来银行获利的损失。因此本文首先建立不同信用等级下客户的流失率和贷款年利率的关系。然后将客户流失率作为“折扣”因子融入至式（4）中调整银行的收益变化。

首先调用 Matalab cftool 工具箱分别选择指数函数（Exponential）、多项式函数（Polynomial）、幂函数（Power）、对不同等级下客户流失率和年贷利率进行拟合，如下表所示：

表 5.7 年贷利率与客户流失率拟合函数表

函数形式	SSE	R-square	RMSE
风险等级 A			
$y=0.77\times e^{1.45x}-2.31e^{-26.48x}$	0.003976	0.998	0.01261
$y=640.9x^3-258.6x^2+37.97x-1.12$	0.004516	0.9977	0.01344
$y=-0.17x^{-0.68}+1.548$	0.004509	0.9978	0.01317
$y=2.41\sin(23.62x-1.446)+1.56\sin(28.6x+1.14)$	0.004429	0.9978	0.01388
风险等级 B			
$y=0.69\times e^{1.98x}-2.07e^{-25.93x}$	0.002722	0.9986	0.01043
$y=552.8x^3-225.1x^2+33.99x-1.02$	0.003314	0.9982	0.01151
$y=-0.297x^{-0.546}+1.721$	0.002836	0.9985	0.01044
$y=2.55\sin(24.51x-1.49)+1.75\sin(28.93x+1.18)$	0.002959	0.9984	0.01134
风险等级 C			
$y=0.71\times e^{1.94x}-1.94e^{-23.71x}$	0.003414	0.9982	0.01169
$y=504.7x^3-207.4x^2+32.16x-0.97$	0.003532	0.9982	0.01189
$y=-0.51x^{-0.43}+2.05$	0.003812	0.998	0.01211
$y=2.93\sin(22.52x-1.42)+2.11\sin(25.94x+1.34)$	0.003669	0.9981	0.01263

如表 5.7 所示各风险等级下年贷利率 x 与客户流失率 y 的拟合函数如上。其中，SSE、R-square 和 RMSE 分别表示误差平方和，确定系数和均方误差其均为反映拟合误差的指标。这里以风险等级 A 时为例，绘制其年贷利率和客户流失率的拟合图像如下：

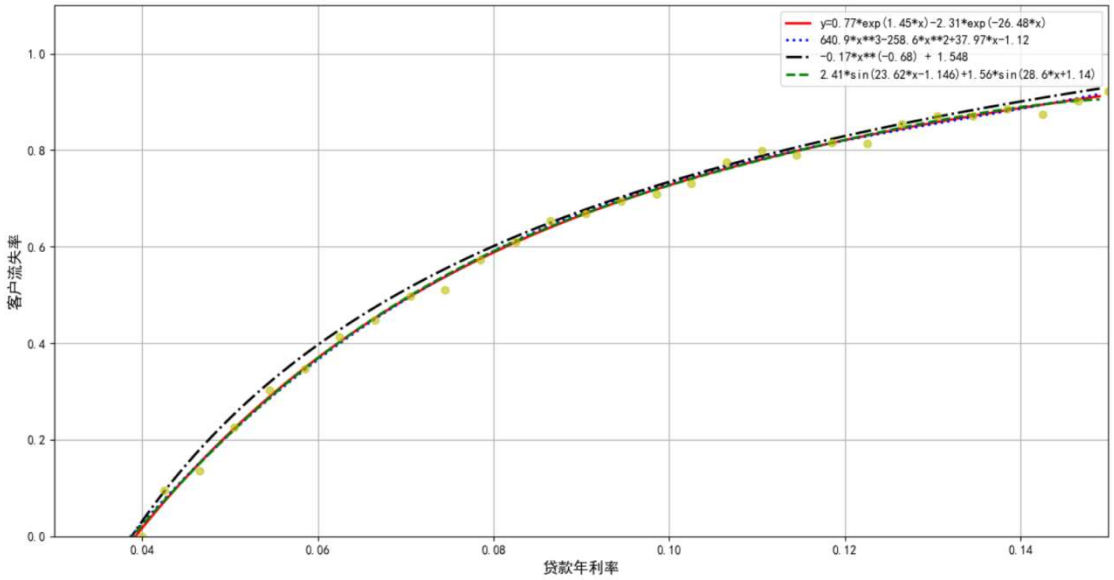

图 5.7 风险等级 A 年贷利率与客户流失率四类函数拟合图

结合表 5.7 和图 5.7 可以看出上述四种函数在不同风险等级下其拟合效果均较好，且各函数拟合效果相差不大。考虑到模型的鲁棒性（抗扰动能力，非病态）和复杂度，本文选择使用简单的三次多项式函数建立年代利率和用户流失率间的关系。

基于上述研究，本文考虑建立客户流失率与银行收益“折扣”因子将的函数关

系，如下：

$$d=-\gamma l_i^2+1, l_i\in[0,1] \quad (5)$$

上式中，d 为用户流失率收益折扣因子；

l_i 为信用等级为 i 时的客户流失率；

γ 为超参数，可以取 1；

考虑将收益“折扣”因子融入至式（4）中，则最后银行考虑顾客违约和顾客流失的信贷策略函数如下：

$$\max \quad \mathrm{Pro}=\sum_{r=1}^{5} n_r \bullet d_r \bullet f_r \bullet (1+\lambda_r m_r)^{t_r} \quad (6)$$

基于线性规划等相关模型最大化式（6）即可获得用户不同信贷风险等级或信誉评级下的借贷策略。

5.2 基于集成学习的企业风险等级识别与信贷总额固定的信贷策略建模

题目要求对于附件二所给企业确定其借贷方案，分析知由于附件二所给企业缺少信誉等级这一关键参数，因此本文首先基于利用附件一所给数据中的销项发票和进项发票信息基于特征工程设计相关特征，然后考虑基于集成学习训练 XGBoost、随机森林等模型实现企业信誉等级的自动识别。根据附件二识别所得的企业信誉等级设计借贷方案。

5.2.1 基于进销发票信息的企业风险等级识别特征工程

企业的进项发票和销项发票信息是企业的核心数据，其直接反映了企业的经营状况变化，因此对这些信息和数据的有效分析可以直接用于指导信贷方案的设计和确定。本文针对附件所给企业的发票信息设计相关特征，如下：

表 5.8 企业经营信息特征指标

特征类型	特征	计算方式
销售类	月开票总额单月同比	（上月开票总额-当月开票总额）/上月开票总额
	近三月开票总额单月同比	（近三月开票总额-3*当月开票总额）/近三月开票总额
	有效客户（产生有效交易）数变化	1-当月有效客户数/上月有效客户数
	下游客户集中度变化	指定周期（一年）下游客户开票总额/当月总开票额
	负数发票单月同比	当月负数发票总额/上月负数发票总额
	作废发票单月同比	当月作废发票总额/上月作废发票总额
支出类	月进销匹配度	1-（当月进项发票总额/当月销项发票总额）
	近三月进销匹配度	1-（近三月进项发票总额/近三月销项发票总额）
	月收票总额单月同比	（上月收票总额-当月收票总额）/上月收票总额
	近三月收票总额单月同比	（近三月收票总额-3*当月开票总额）/近三月收票总额

如表 5.8 所示，基于销项与进项发票信息本文首先设计两大类特征，即销售类和支出类。同时针对这两大类特征设计如月开票总额单月同比、月进匹配度等指标。考虑到

附件二中给出不同企业的一年发票信息，因此本文针对时序数据进行“开窗”处理，窗口大小由上述设计特征知为一个月或三个月。由滑动窗口计算取平均值得到该特征的特征值，如下：

$$\overline{Fe_i}=\frac{1}{n}\sum_{j=1}^{n}Fe_{ij} \tag{7}$$

上式中，Fe_{ij}即为特征Fe_i在第j个窗口下的取值，若按每月“开窗”滑动处理则考虑一年周期$n=11$，若窗口大小为3月则$n=9$。基于上述特征训练模型实现企业信誉等级的自动识别。

5.2.2 基于集成学习的企业信誉自动识别

集成学习[8]在最初的研究中其出发点主要是关注弱学习器。一般可以认为弱学习是指其分类准确率略高于随机猜测的分类器，即对于二分类问题弱学习器的准确率略高于50%。然而，在实际情况中通常希望利用较少的分类器获得较高的准确率，因此一般利用较强的分类器进行集成。

（1）同质学习和异质学习

①同质学习：个体学习器类型相同。如分类器全为C4.5决策树或神经网络等。

②异质学习：个体学习器的类型不同。如分类器类型既包括有决策树同时又包括有神经网络等。

一般情况下，由于不同类型分类器的数学理论、思想和工作机制往往不同，因此很难全面比较分类器性能的好坏，故集成学习大多是研究同质学习。然而在实际中，我们也经常将多个不同的分类结果较好的异质学习器的最终分类结果进行集成，以实现分类结果精度的提高。这主要是因为，不同类型的分类器由于其分类机理不同，将其结合时其结果往往能相互补充。此外，根据embedding的思想，构建深度神经网络模型，学习特征向量，在将其作为树分类器的特征输入以提高精度也是常见的策略。

（2）集成策略

针对结合策略的不同，主要可以分为平均法、投票法和学习法。

①平均法（回归问题）：平均法包括简单平均和加权平均。简单平均是指将各学习器的分类结果取平均，此时个分类器权值相等，即各分类器的重要程度相同。加权平均各分类器的权值往往通过学习所得，如AdaBoost[9]，准确率较高的分类器，其权值一般也较大。然而，由于数据的不充分或噪声的影响，大规模集成时较易出现过拟合的现象。因此，一般而言当个体分类器性能相差较大时，宜采用加权平均法，而个体分类器性能相差不大时，简单平均法往往效果较好。

②投票法（分类问题）：投票法包括绝对数投票法、相对多数投票法和加权投票法。绝对数投票法是指当得票数超过一定阈值时则预测为该标记，例如投票数超过半数时则记为正类，否则为负类。相对多数投票法即为少数服从多数。加权投票法与加权平均法类似，

③学习法（Stacking）：学习法是指利用另一个学习器将分类器结果进行结合，这里个体学习器可称为初级学习器，而将结合学习器称为次级学习器或元学习器。其中Stacking为学习法的典型代表。在实际中，一般利用交叉训练产生次级学习器的训练集以减缓过拟合的影响。根据相关研究表明，将初级学习器的输出类概率作为次级学习器的输入属性，采用多响应线性回归（Multi-response Linear Regression, MLR）作为次级分类器效果较好。

（3）串行集成与并行集成

针对集成方法的不同，主要可以分为并行集成和串行集成。

①串行集成：个体学习器间存在强依赖的关系，后一个学习器的生成需依赖前一个学习器的结果。其代表算法为Boosting，包括AdaBoost、GBDT、XGBoost等。

②并行集成：个体学习器间不存在依赖关系，可以同时训练多个学习器，适合分布式并行计算。其代表算法为Bagging[10]、随机森林[11]。

集成学习追求“好而不同”，即每个个体分类器在保证较高的分类准确率的同时，又能有较大的差异，这样集成结果能更好的反映样本的整体特征。事实上近年来KDD、Kaggle等有关数据挖掘的竞赛题冠军组无一例外采用了集成学习的方法，其有些取胜算法使用超过几十种模型的集成。

这里假设有k个基学习器，每个学习器的分类误差为ε_i，其反映每个分类器的准确度，该误差服从均值为$E(\varepsilon_i)=0$，方差为$E(\varepsilon_i^2)=v$，协方差为$E(\varepsilon_i\varepsilon_i)=c$的多维正态分布，其中协方差即反映各个分类器的差异，则集成模型的分类平方误差期望为：

$$E[(\frac{1}{k}\sum_i\varepsilon_i)^2]=\frac{1}{k^2}E[\sum_i(\varepsilon_i^2+\sum_{j\neq i}\varepsilon_i\varepsilon_j)]=\frac{1}{k^2}E[\sum_i\varepsilon_i^2]+\frac{1}{k^2}E[\sum_i\sum_{j\neq i}\varepsilon_i\varepsilon_j]$$
$$=\frac{1}{k^2}kv+\frac{1}{k^2}\frac{k-1}{k}c=\frac{1}{k}v+\frac{k-1}{k}c \quad (8)$$

从上式可以看出，一般情况下，在基学习器差异较大时，即分类错误完全不相关（$c=0$）时,可以认为集成平方误差的期望随着集成规模的增大而线性减少，即集成学习往往比单个分类器的效果要好，且随这个体学习器分类差异的增加，集成学习准确度也将增加。理想情况下，当k趋于无穷且基学习器相互独立时其误差将趋于0，然而在实际情况下基学习器不会完全独立，故其误差不会为0。

（4）随机森林

随机森林是Bagging算法即并行集成学习的著名代表，Bagging主要思想是通过拔靴法（有放回抽样）随机抽取多个样本集同时训练多个分类器，进行集成。如图5.8所示：

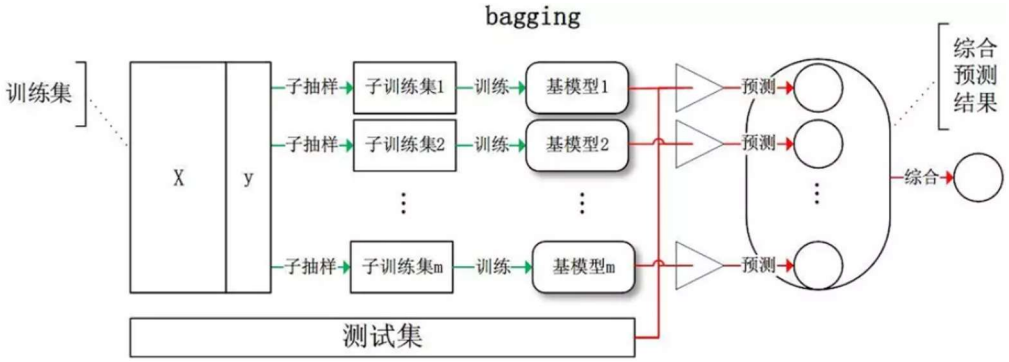

图 5.8 Bagging 框架

其中随机样本集的生成一般采用拔靴法即自助采样法，即有放回的随机采样，当重复进行多次采样操作后，将有 36.8%的样本没有被抽取到，其可以作为验证集。在多个分类器得到预测结果后，对于分类问题可以采用集成法得到最终结果，而对于回归问题可采用平均法得到最终结果，其算法步骤如下所示：

模型输入：训练集 $D=\{(x_1,y_1),(x_2,y_2),...,(x_m,y_m)\}$，基学习器 L，训练轮数 T

模型输出：$H(x)=\arg\max_{y\in Y}\sum_{t=1}^{T}\prod(h_t(x)=y)$

1. for t=1,2,…,T do
2. $h_t=L(D,D_{bs})$
3. end for

Bagging 算法效率较高，其时间复杂度与训练基学习器同阶。此外 Bagging 能够不经修改的适用于回归和多分类等问题。

随机森林则是在样本随机选择的同时，多棵决策树的构建过程中每棵决策树的属性的选择也随机，然后根据选择属性构建最优决策树进行集成。这样基学习器的多样性不仅来自于样本的扰动，同时还来自属性的扰动，也就是说，随机森林的随机性主要体现在两个方面：①样本的随机选择，②属性的随机选择。因此随机森林最终集成的泛华性能和个体学习器的差异进一步加大。此外，由于随机森林每棵树的训练都是独立的，其并不依赖前一棵树的分类结果，因此随机森林天生就适应并行，故其训练效率和模型的泛化误差往往均优于 Boosting，然而该模型的精度较低，一般情况下低于 Boosting 方法几个百分点。其模型结构如下图所示：

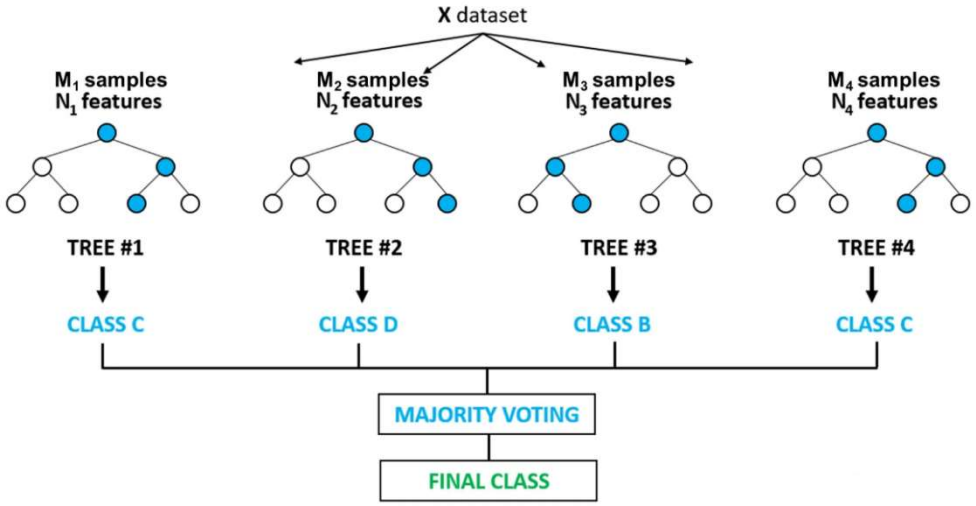

图 5.9 随机森林模型示意图

如图 5.9 所示，随机森林首先从原始数据集 X 中随机抽取部分数据，同时选择不同特征训练多个基学习器，如分类回归决策树 CRDT。然后，利用多个基学习器对样本进行分类，并利用投票法获得最终分类结果。

（5）XGBoost

Boosting 算法簇的主要代表包括 AdaBoost、GBDT、XGBoost 及其改进等。其工作机制大致为：

Step1. 从初始训练集中训练出一个基学习器；

Step2. 根据基学习器的性能表现调整样本分布，使得分类错误的样本在后续的训练中得到更多的关注；

Step3. 基于调整后的样本调整训练下一个基学习器；

Step4. 重复步骤 step1、step2 直至基学习器达到指定数目；

Step5. 对基学习器预测结果进行加权融合，输出。

XGBoost 可视为线性加法模型，其采用增量学习的方法构建 K 棵分类回归决策树，这里假定分类数据集为 $D=\left(x_i,y_i\right)$，则其对样本 x_i 的预测值如式（9）：

$$\hat{y}=\phi\left(x\right)=\sum_{k=1}^{K}f_k\left(x_i\right),\qquad f_k\in F \tag{9}$$

式（9）中 f_k 即为 CART 树，F 为假设空间，XGBoost 的目标函数如式（10）所示：

$$\begin{aligned}&L\left(\phi\right)=\sum_{i=1}^{n}l\left(y_i,\hat{y}^{(t)}\right)+\Omega\left(f_t\right)=\sum_{i=1}^{n}l\left(y_i,\hat{y}^{(t-1)}+f_t\left(x_i\right)\right)+\Omega\left(f_t\right)\\&f\left(x\right)=w_{q(x)}\\&\Omega\left(f\right)=\gamma T+\frac{1}{2}\lambda\left\|w\right\|_2\end{aligned} \tag{10}$$

上式中，$q\left(x\right)$ 表示将样本 x 分配到叶子节点上，而 w 表示该叶子节点的大分，因此 $w_{q(x)}$ 可以表示该样本的预测值。因此可以看到第 t 轮迭代的预测值 $\hat{y}^{(t)}$ 即为前一轮的预测值 $\hat{y}^{(t-1)}$ 与本次迭代的预测值 f_t 共同组成。同时为避免训练函数过拟合，这里添加 $\Omega\left(f\right)$，

为目标函数的正则化项。其中T为叶子节点数目，用于表示树的复杂程度，γ,λ分别表示权值，为超参数。我们计算 Jacobian 矩阵和 Hessian 矩阵，并利用泰勒公式对式（4）进行二阶展开，同时对叶节点进行分裂并计算叶节点分裂前后的增益以确定最优划分属性，构建 GBDT 树，多轮迭代得到最终结果。

本文考虑使用决策树，随机森林和 XGBoost 对其进行分类。首先，以附件一作为原始数据，按 9:1 分为训练集和测试集。拆分后训练集样本数目为 110，测试集样本数目为 13。

模型评价指标选择准确率（precision）、召回率（recall）和 F1 值，其计算公式如下：

$$P=\frac{TP}{TP+FP}$$
$$R=\frac{TP}{TP+FN} \qquad (11)$$
$$F1=\frac{2P\bullet R}{P+R}$$

式（11）中 TP、FP、FN 分别代表真正例（实际为正类，判定为正类）、假正例（实际为负类，判定为正类）、假负例（实际为正类，判定为负类）这三种分类情况。

各模型超参数的设置如下：

①决策树：数深度为 20，min_impurity_split=0.1；

②随机森林：基学习数目为 250，min_impurity_split=0.1；

③XGBoost：基学习器数目为 150，learning_rate=0.15

最终各模型的实验结果如下：

表 5.9 基于集成学习的企业信誉等级自动识别结果

模型名称	Precision	Recall	F1
训练集			
决策树	1	1	1
随机森林	1	1	1
XGBoost	1	1	1
测试集			
决策树	0.68	0.59	0.632
随机森林	0.71	0.66	0.684
XGBoost	0.74	0.71	0.725

如表 5.9 所示，可以看出上述三类模型在训练集上其准确率、召回率和 F1 值均到达 1，而在测试集上其各指标均有一定的下降，表明该模型存在一定的过拟合风险。

5.2.3 固定金额下的信贷策略建模

基于上述识别结果分析知附件二中 302 家企业，绘制其信誉等级分布如图 5.10 所示：

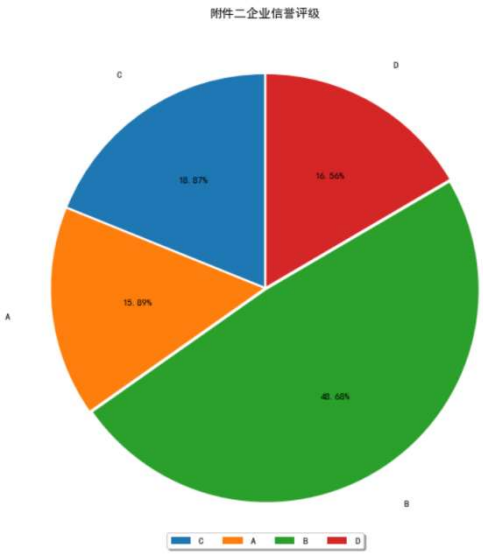

图 5.10 附件二企业信誉等级分布饼图

由图 5.10 可以看出附件二中 302 家企业中信誉等级为 B 的企业占种企业数的 48.68%，其它三种企业的信誉等级分别保持在 16%左右。由题目知该银行的年度信贷总额为 1 亿元，则将其带入式（5）求解得不同信誉等级企业的贷款额度及贷款年利率如下所示：

表 5.10 不同信誉或风险等级下企业的信贷方案

信誉评级	风险等级	贷款额度（万元）	贷款年利率
A	低	58.27（50～70）	5.2%（4%～6%）
B	中	40.79（35～50）	6.3%（6%～8%）
C	高	17.48（15～20）	10.4%（8%～12%）
D	极高	5.82（0，10）	12.3%（10%～15%）

不同信誉或风险等级的企业贷款方案如表5.10所示，其中括弧外的结果为由式(5)求解所得的标准贷款金额和利率，而括弧内的结果为调整超参数后的建议贷款方案选择区间。其中对于信誉评级为 D 的企业其建议贷款金额为 5.82 万元，但考虑到实际题目要求其贷款金额需大于等于 10 万元，因此该值表示含义为部分信用评级为 D 的企业将无法获得贷款。而获得贷款的信誉等级为 D 的企业其贷款金额也只能是 10 万元。

5.3 突发事件下考虑企业违约的银行借贷策略建模

由于某些突发因素如新冠病毒等将造成企业经营管理的不确定性和风险[12][13]。同时不同规模和信誉的企业在面对突发因素时其受到的影响也将不同。参考 5.1 小结相关分析，假设突发因素将影响企业的违约率 f_r，同时规模越小、信誉等级越低的企业其违约率将越高，反之，规模越大、信誉等级越高的企业其违约率将越低。因此，本文首先考虑企业信誉、规模建立衡量企业抗打击能力的“可靠性”模型[14][15]，然后建立企业“可靠性”与违约率间的函数关系。最后考虑实际情况中的不确定性，指定违约率服从 Gaussian 分布，求解式（4）获得不同信誉等级企业的借贷策略。

另一方面，商业银行设计的的借贷策略在追求所企业违约可能性最低的情况下，同

时还应考虑帮助微小企业渡过难关。此时可以牺牲部分利益，降低年贷款利率实现纾困帮扶。故该方安案仅考虑贷款金额以期最小化企业违约可能，同时还考虑帮助微小企业生存，设计较低的年利率照顾客户。

（1）企业“可靠性”与违约率函数关系的建立

本文考虑企业抗打击的“可靠性”主要由企业的信誉和规模决定，具体来说即由企业近三月开票总额单月同比和信誉评级这两个指标确定，定义企业“可靠性”指标为 Ra，其计算公式如下：

$$\begin{aligned} Ra_i &= w_r M_i + (1-w_r) R_i \\ M_i &= \frac{\varpi_i}{\max(\varpi_i)} \end{aligned} \tag{12}$$

上式中，M_i, R_i 分别表示 i 企业规模得分和企业信誉得分(归一化)；

ϖ 为企业 i 近三月开票总额单月同比；

w_r 为权值，取值为 0.4；

本文建立企业可靠性 Ra 与违约率 f_r 间的函数关系如下式：

$$f_r = e^{-\beta Ra} - \delta \tag{13}$$

上式中，β，δ 均为超参数，这里取 β=3，δ=0.2；

（2）基于变违约率的借贷方案设计

如式（13）所示，当企业“可靠性” Ra 确定时其违约率 f_r 将唯一确定。然而考虑到实际情况中的不确定性，本文考虑违约率为服从高斯分布的变量，即 $f_r \sim Gaussian(\overline{f_r}, \mu)$，其中 $\overline{f_r}$ 由式（13）确定，μ 值可根据突发事件的严重程度进行调整，这里取 0.05，如下图所示：

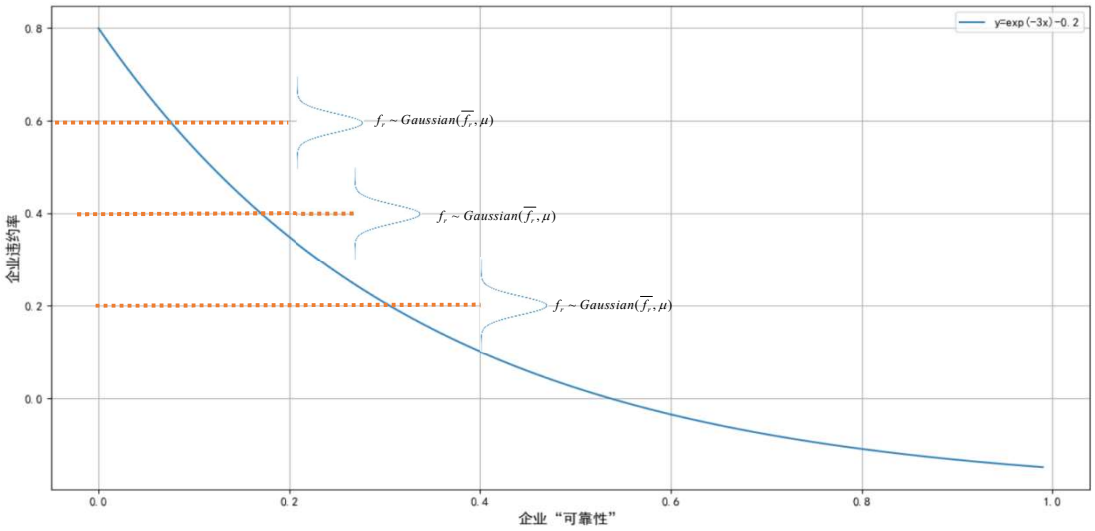

图 5.11 企业“可靠性”与违约率函数图像

最后结合式（13）式（4）获得不同信誉等级、规模下企业的借贷策略如下表所示：

表 5.10 不同信誉和规模下企业的信贷方案

信誉评级	企业规模	贷款额度（万元）	贷款年利率
A	5	73.23（65～80）	4.2%（4%～4.5%）
	4	64.29（60～65）	4.7%（4.5%～5%）
	3	58.02（55～60）	5.2%（5%～5.5%）
	2	51.75（50～55）	5.8%（5.5%～6%）
	1	42.18（40～50）	6.3%（6%～6.5%）
B	≥4	54.63(50～60)	6.7%（6.5%～7%）
	3	42.19(30～50)	7.3%（7%～7.5%）
	≤2	22.13(10～30)	7.8%（7.5%～8%）
C	≥4	22.30（20～25）	8.2%（8%～8.5%）
	3	17.26(15～20)	8.7%（8.5%～9%）
	≤2	11.34(10～15)	9.4%（9%～9.5%）
D	—	0	0

如表 10 所示，在考虑银行收益最大化即最小违约发生的前提下同时降低年利率实现让利，帮助企业抵御风险。不同企业规模和信誉下的信贷方案如表 5.10 所示，其中对于信誉为 D 的企业不考虑提供贷款服务。

6. 敏感性分析

本文考虑对问题二中企业信誉评级自动识别模型的超参数进行敏感性分析，如下：

（1）决策树深度

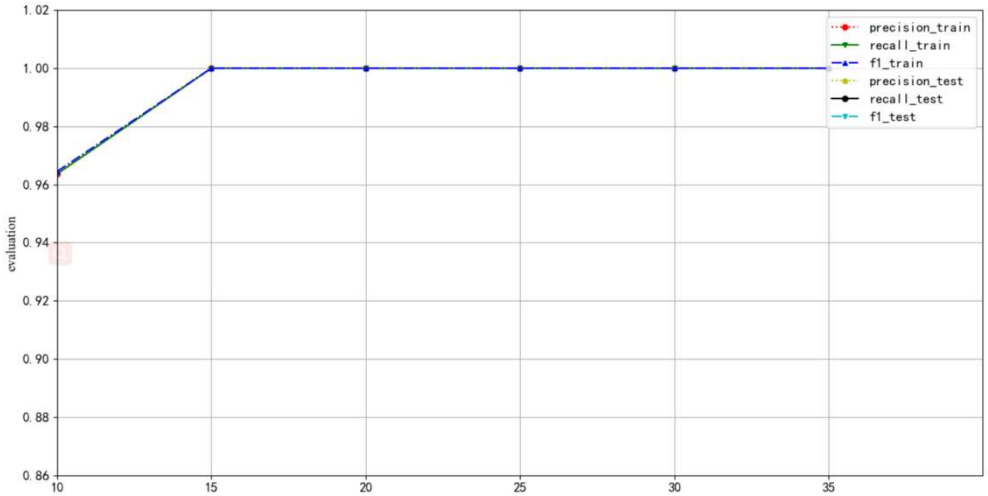

图 6.1 决策树模型深度超参数敏感性分析

如图 6.1 所示，可以看到当决策树模型的树深度超过 15 后其训练集的准确率、召回率和 f1 值均到达 1,即模型对该参数不敏感。

（2）随机森林基学习器数目

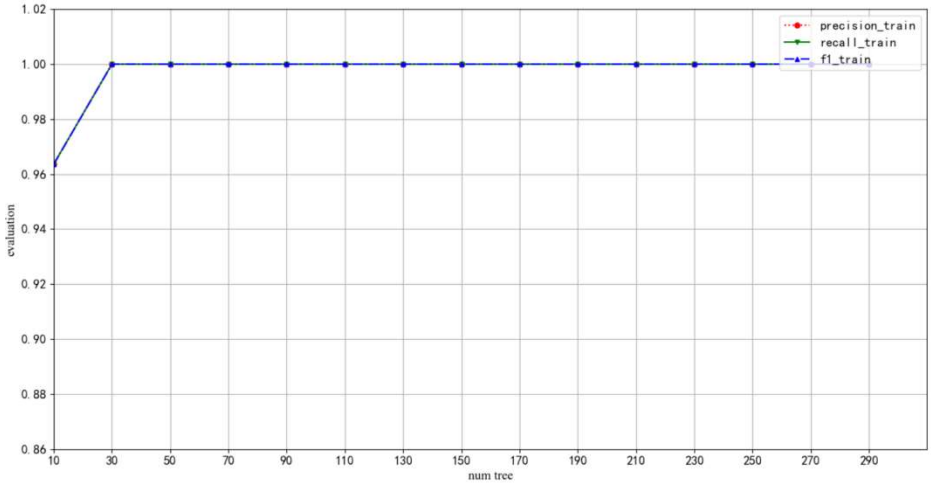

图 6.2 随机森林模型决策树数目超参数敏感性分析

如图 6.1 所示，可以看到当随机森林模型决策树数目超过 20 后其训练集的准确率、召回率和 f1 值均到达 1，即模型对该参数不敏感。

（3）XGBoost 学习率

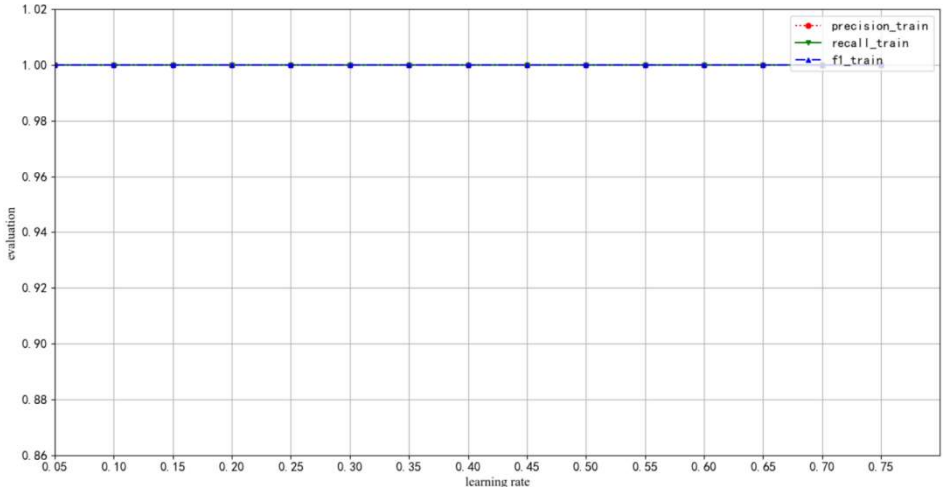

图 6.3 XGBoost 模型学习率超参数敏感性分析

如图 6.3 所示，可以看到学习率的变化对 XGBoost 模型训练集的准确率、召回率和 f1 值无较大影响，即模型对该参数不敏感。

7. 模型的评价与推广

7.1 模型评价

7.1.1 问题一模型评价

问题一分别考虑企业信用和企业发展前景对企业借贷风险进行量化评价。同时在建立企业借贷方案模型时同时考虑借贷年利率与客户流失率的关系以及企业信用等级与违约风险间的关系，建立数学模型对不同信誉评级的企业进行借贷方案的设计，考虑全面。但是该方案未能融入企业进销发票信息，仍需进一步补充完善。

7.1.2 问题二模型评价

问题二首先利用企业进销发票信息设计特征，然后选择集成学习基学习器决策树算

法、随机森林和 XGBoost 对企业信誉评级进行自动分类。最后结合问题一相关模型实现企业信贷策略设计。该方法能基于企业进销发票信息有效实现企业信誉等级的自动识别，但识别精度仍需进一步提升。

7.1.3 问题三模型评价

问题三分析新冠疫情对企业生产活动的影响，综合考虑银行获利以及企业纾困这两方面，在银行收益最大化的同时尽量减少企业负担。对此，考虑企业历史经营状态，设计企业抗打击能力指标——企业“可靠性”，同时假设企业“可靠度”越高，违约率将越低，建立企业“可靠性”与企业违约率间的函数关系。最后，考虑到现实世界中的不确定性，对企业的违约率指定为 Gaussian 分布，实现在突发因素影响下企业借贷方案的动态调整。该模型能有效设计不同规模企业在突发因素影响下的借贷方案，模型具有较强的适应性。

7.2 模型推广

7.2.1 问题一模型推广

问题一考虑企业的流失率和企业的违约率设计借贷方案模型可以推广至任意产品的销售策略和定价方案的制定中，即产品售价越高，潜在客户的流失率将会上升。

7.2.2 问题二模型推广

问题二中对于企业信誉的评价模型如 XGBoost、随机森林、可以推广至其它任意的分类问题中，如潜在客户类型的识别，基于各类指标的故障识别和疾病识别等。

7.2.1 问题三模型推广

问题三对企业抗打击能力即“可靠性”的定义可以推广至其它领域，如供应链的可靠性，大型复杂网络的可靠性，服务的“可靠性”等等。

8. 参考文献

[1] 林华. 基于交易平台的供应链金融产品的风险管理研究[D].浙江工业大学,2017.

[2] 中证指数有限公司. http://www.csindex.com.cn/zh-CN/indices/index?class_1=1&class_18=18&class_10=10&class_7=7&is_custom_0=1.

[3] 宋学彦. 供给侧改革下我国造纸行业市场集中度研究[D].南京林业大学,2019.

[4] 伍梓维. 2007-2016 年我国旅行社产业集中度演化及影响因素分析[D].湘潭大学,2019.

[5] 国泰君安-来自 130 个公司的全透视：行业集中度，在提升吗，会延续吗-191011. http://m.hibor.com.cn/wap_detail.aspx?id=0279e7cd52c729cce09a861866516ec1.

[6] GB/T 4754-2017，国民经济行业分类标准[S].

[7] 陈晓瑞. XY 银行信贷放款管理研究[D].吉林大学,2018.

[8] Zhou Z H. Ensemble Methods: Foundations and Algorithms[M]. Taylor & Francis, 2012.

[9] Yoav Freund, Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting[M]// Computational Learning Theory. Springer Berlin Heidelberg, 1995:119-139.

[10] Breiman L. Bagging predictors[M]. Kluwer Academic Publishers, 1996.

[11] Breiman L. Random forests, machine learning 45[J]. Journal of Clinical Microbiology, 2001, 2: 199-228.

[12] 陶有珠. 商业银行抗突发事件能力评价与提高[D].合肥工业大学,2009.

[13] 陆嘉. 我国城市突发事件下银行业资金动员的供求研究[D].南京航空航天大学,2005.

[14] 吴启理. 企业战略承诺可靠性对企业价值的影响研究[D].内蒙古大学,2019.

[15] Snyder L V. Supply chain robustness and reliability: Models and algorithms[D]. Northwestern University, 2003.

附录

附录一：数据分析代码

```
import pandas as pd
import matplotlib.pyplot as plt
from pylab import *
import wordcloud
import datetime
import random
import collections
import jieba
import imageio
import matplotlib as mpl
import seaborn as sns
import numpy as np
from collections import Counter
import random
import math
from pandas.tools.plotting import parallel_coordinates

mpl.rcParams['font.sans-serif'] = ['SimHei']


def industry_plt(dict_main):
    all_colors = list(plt.cm.colors.cnames.keys())
    c = random.choices(all_colors, k=len(dict_main.values()))
    # Plot Bars
    fig = plt.figure(figsize=(16, 10), dpi=80)
    ax1 = fig.add_subplot(111)
    list_all = list(dict_main.values())
    list_count = []
    list_prec = []
    list_prec_sum = [0]

    for i in list_all:
        list_count.append(int(i))
        list_prec.append(float(i / sum(list_all)))
        list_prec_sum.append(min(round(list_prec_sum[-1] + list_prec[-1], 2), 1.0))
    print(list_prec_sum)
    list_prec_sum.pop(0)
    plt.bar(dict_main.keys(), list_count, color=c, width=.5)
    for i, val in enumerate(list_count):
```

```
            plt.text(i, val, float(val), horizontalalignment='center', verticalalignment='bottom',
                     fontdict={'fontweight': 500, 'size': 12.5})
    # Decoration
    plt.gca().set_xticklabels(dict_main.keys(), rotation=20, horizontalalignment='right', fontsize=12.5)
    plt.title("各企业所属行业类别分布", fontsize=12.5)
    ax1.set_ylabel('各企业所属行业类别数目', fontsize=12.5)
    ax1.set_yticks(np.arange(0, math.ceil(list_count[0] / 100)), 100)

    ax2 = ax1.twinx()
    ax2.plot(list_prec_sum)
    ax2.scatter([i for i in range(len(list_prec_sum))], list_prec_sum, s=20, marker='x', color='red')
    # ax2.plot()
    ax2.grid(True)
    # 设置数字标签
    for a, b in enumerate(list_prec_sum):
        plt.text(a, b, b, ha='center', va='bottom', fontsize=12.5)
    ax2.set_yticks(np.arange(0, 1.1, 0.1))  # 设置右边纵坐标刻度
    ax2.set_ylabel('各企业所属行业类别数目占比', fontsize=12.5)
    plt.show()


def cloud_img(comps, img):
    cut_list = []
    for comp_temp in comps:
        cut_text = jieba.cut(comp_temp, cut_all=False)
        cut_list += cut_text
    cut_name = dict(collections.Counter(cut_list))
    cut_name_dict = {}
    for cut_name_temp in cut_name:
        if cut_name[cut_name_temp] > 1:
            cut_name_dict[cut_name_temp] = cut_name[cut_name_temp]
    print(cut_name_dict)

    # wordcloud_2 = wordcloud.WordCloud(font_path='C:\Windows\Fonts\STZHONGS.TTF', mask=img,
    #                                   background_color='white',
    #                                   prefer_horizontal=0.75, random_state=50).fit_words(cu
t_name_dict)
    # plt.figure("company name cloud")
    # plt.imshow(wordcloud_2)
    # plt.axis('off')
    # plt.show()
```

```
def comp_any(raw_data_comp):
    name_comp = raw_data_comp['企业名称']
    name_comp_clean = []
    for name_temp in name_comp:
        name_temp = name_temp.replace('*', '')
        if '有限责任公司' in name_temp:
            name_temp = name_temp.replace('有限责任公司', '')
        if '有限公司' in name_temp:
            name_temp = name_temp.replace('有限公司', '')
        if '分公司' in name_temp:
            name_temp = name_temp.replace('分公司', '')
        if '公司' in name_temp:
            name_temp = name_temp.replace('公司', '')
        name_comp_clean.append(name_temp)
    image = imageio.imread(r'money.jpg')
    cloud_img(name_comp_clean, image)


def plot_pie(dicts, name_title):
    labels = list(dicts.keys())
    X = list(dicts.values())
    fig = plt.figure(figsize=(24, 16), dpi=80)
    explode = tuple([0.01] * len(X))
    plt.pie(X, explode=explode, labels=labels, labeldistance=1.2, autopct='%1.2f%%', shadow=False, s
tartangle=90,
            pctdistance=0.6)
    plt.legend(loc='lower center', ncol=5, fancybox=True, shadow=True)
    plt.title(name_title, fontsize=12.5)
    plt.show()


def read_data(path_raw):
    raw_data_comp = pd.read_excel(path_raw, encoding='utf_8_sig', sheet_name='企业信息')
    # comp_any(raw_data_comp)
    # plot_pie(dict(Counter(raw_data_comp['信誉评级'])), '信誉评级')
    plot_pie(dict(Counter(raw_data_comp['是否违约'])), '是否违约')


def func_deg():
    x = np.arange(0, 5.01, 0.01)
    y_1 = -x / 5 + 1
    y_2 = cos(np.pi / 10 * x)
    y_3 = 1 / (x + 1)
```

```
    y_4 = 1 / (1 + exp(x - 2.5))
    plt.plot(x, y_1, label="y=-x/5+1", color="red", linewidth=2, linestyle='-')
    plt.plot(x, y_2, label="y=cos(pi*x/10)", color="blue", linewidth=2, linestyle=':')
    plt.plot(x, y_3, label="y=1/(x+1)", color="black", linewidth=2, linestyle='-.')
    plt.plot(x, y_4, label="y=1/(1+exp(x-2.5))", color="green", linewidth=2, linestyle='--')
    plt.grid(True)
    plt.xlim(0, 5)
    plt.ylim(0, 1)
    plt.xlabel('风险打分', fontsize=12.5)
    plt.ylabel('违约率', fontsize=12.5)
    plt.legend()
    plt.show()


def loss_fig(path_loss):
    raw_data_comp = pd.read_excel(path_loss, encoding='utf_8_sig', sheet_name='Sheet1')
    print(raw_data_comp)
    x_point = list(raw_data_comp['贷款年利率'])[1:]
    y_point = list(raw_data_comp['客户流失率'])[1:]
    x = np.arange(0.03, 0.15, 0.001)
    y_1 = 0.7701 * exp(1.453 * x) - 2.306 * exp(-26.48 * x)
    y_2 = 640.9 * x ** 3 - 258.6 * x ** 2 + 37.97 * x - 1.12
    y_3 = -0.17 * x ** (-0.68) + 1.548
    y_4 = 2.405 * sin(23.62 * x - 1.446) + 1.559 * sin(28.6 * x + 1.141)
    plt.plot(x, y_1, label="y=0.77*exp(1.45*x)-2.31*exp(-26.48*x)", color="red", linewidth=2, linestyle='-')
    plt.plot(x, y_2, label="640.9*x**3-258.6*x**2+37.97*x-1.12", color="blue", linewidth=2, linestyle=':')
    plt.plot(x, y_3, label="-0.17*x**(-0.68) + 1.548", color="black", linewidth=2, linestyle='-.')
    plt.plot(x, y_4, label="2.41*sin(23.62*x-1.146)+1.56*sin(28.6*x+1.14)", color="green", linewidth=2, linestyle='--')
    plt.plot(x_point, y_point, 'yo', alpha=0.6, linewidth=0.1)
    plt.grid(True)
    plt.xlim(0.03, 0.15)
    plt.ylim(0, 1.1)
    plt.xlabel('贷款年利率', fontsize=12.5)
    plt.ylabel('客户流失率', fontsize=12.5)
    plt.legend()
    plt.show()


def bar_any(dict_main):
    all_colors = list(plt.cm.colors.cnames.keys())
```

```
    c = random.choices(all_colors, k=len(dict_main.values()))
    # Plot Bars
    fig = plt.figure(figsize=(16, 10), dpi=80)
    ax1 = fig.add_subplot(111)

    list_count = dict_main.values()

    num = [str(i) for i in list(dict_main.keys())]

    plt.bar(dict_main.keys(), list_count, color=c)
    for i, val in enumerate(list_count):
        plt.text(i, val, float(val), fontdict={'fontweight': 12.5, 'size': 12.5})
    # Decoration
    plt.title("企业不同信誉等级数目柱状图", fontsize=12.5)
    ax1.set_ylabel('各类信誉等级企业数目', fontsize=12.5)

    plt.show()



def pre_analysis(path_raw):
    raw_data = pd.read_csv(path_raw, encoding='utf_8_sig', engine='python')

    # plot_pie(dict(Counter(list(raw_data['信誉评级']))), '附件二企业信誉评级')
    bar_any(dict(Counter(list(raw_data['信誉评级']))))


def main():

    path = r'../../../附件 2：302 家无信贷记录企业的相关数据.xlsx'
    path_out = r'../../../附件二销项发票.csv'
    path_in = r'../../../附件二进项发票.csv'

    # index_system_out(path_out, r'../../../index_2.csv')

    # index_system_out(path, path_in)
    # index_system(path_in)
    # index_system_in(path_in, r'../../../index_add_2.csv')
    # pie_chart(path)

    # read_data(path)
    # path_loss = path = 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'
    # loss_fig(path_loss)
```

```
    # func_deg()
    # comp_dict = {'公共管理': 10, '建筑业': 20, '商务服务业': 11, '其它': 9, '科学研究和技术服务业': 15, '批发和零售业': 15, '交通运输、仓储和邮政业': 4,
    #  '农、林、牧、渔业': 8, '住宿和餐饮业': 5, '制造业': 14, '文化、体育和娱乐业': 6, '教育': 6}
    # comp_dict_sort = {}
    # for i,j in sorted(comp_dict.items(), key=lambda item:item[1], reverse=True):
    #       comp_dict_sort[i] = j
    # industry_plt(comp_dict_sort)


if __name__ == "__main__":
    main()
```

附录二：信誉评级识别源代码

```
import pandas as pd
import random
from sklearn import tree
from sklearn.metrics import accuracy_score, confusion_matrix, recall_score, f1_score, roc_curve, auc
import numpy as np
from sklearn import ensemble
import itertools
import xgboost as xgb
from sklearn import tree
from subprocess import check_call
# import pydotplus
import codecs
import itertools
import matplotlib.pyplot as plt


plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号#有中文出现的情况，需要 u'内容'


def plot_weight(acc_train, recall_train, f1_train, acc_test, recall_test, f1_test):

    # weight_x = [i for i in np.arange(0.01, 0.3, 0.01)]
    weight_x = [i for i in np.arange(0.05, 0.8, 0.05)]
    plt.plot(weight_x, acc_train, color='r', linestyle=':', marker='o', markerfacecolor='r', markersize=5,
             label='precision_train')
    plt.plot(weight_x, recall_train, color='g', linestyle='-', marker='v', markerfacecolor='g', markersize=5,
```

```
                label='recall_train')
    plt.plot(weight_x, f1_train, color='b', linestyle='-.', marker='^', markerfacecolor='b', markersize=5,
                label='f1_train')
    # plt.plot(weight_x, acc_test, color='y', linestyle=':', marker='p', markerfacecolor='y', markersize=
5,
    #             label='precision_test')
    # plt.plot(weight_x, recall_test, color='k', linestyle='-', marker='o', markerfacecolor='k', markersiz
e=5,
    #             label='recall_test')
    # plt.plot(weight_x, f1_test, color='c', linestyle='-.', marker='v', markerfacecolor='c', markersize=
5,
    #             label='f1_test')
    plt.xticks(np.arange(0.05, 0.8, 0.05), fontsize=12.5)
    plt.xlim((0.05, 0.8))
    plt.yticks(np.arange(0.86, 1.02, 0.02), fontsize=12.5)
    plt.ylim((0.86, 1.02))
    plt.grid(True)
    plt.legend(loc=1, fontsize=12.5)
    font2 = {'family': 'Times New Roman',
             'weight': 'normal',
             'size': 12.5}
    plt.xlabel('learning rate', font2)
    plt.ylabel('evaluation', font2)
    plt.show()


def data_split(raw_data):
    a = [i for i in range(len(raw_data))]
    random.shuffle(a)
    train_data = raw_data.iloc[a[0:int(0.9*len(a))]]
    test_data = raw_data.iloc[a[int(0.9*len(a)):]]
    train_data.to_csv(r'../../../train.csv', encoding='utf_8_sig', index=False)
    test_data.to_csv(r'../../../test.csv', encoding='utf_8_sig', index=False)


def DT(data_train, label_train, data_test, label_test):
    acc_train = []
    recall_train = []
    f1_train = []
    aucs_train = []
    acc_test = []
    recall_test = []
    f1_test = []
```

```
    aucs_test = []

    # for i in range(10, 101, 10):
    for temp in range(10, 40, 5):
        clf = tree.DecisionTreeClassifier(max_depth=temp, min_impurity_split=0.1)
        decision_tree = clf.fit(data_train, label_train)
        train_predict = decision_tree.predict(data_train)

        acc_train.append(accuracy_score(label_train, train_predict))
        recall_train.append(recall_score(label_train, train_predict, average='weighted'))
        f1_train.append(f1_score(label_train, train_predict, average='weighted'))

        test_predict = decision_tree.predict(data_test)
        test_predict_prob = decision_tree.predict_proba(data_test)
        acc_test.append(accuracy_score(label_test, test_predict))
        recall_test.append(recall_score(label_test, test_predict, average='weighted'))
        f1_test.append(f1_score(label_test, test_predict, average='weighted'))
        print(test_predict_prob)
        one_hot = []
        for i in label_test:
            temp = [0] * 4
            temp[i] = 1
            one_hot += temp
        fpr, tpr, _ = roc_curve(one_hot, list(itertools.chain.from_iterable(test_predict_prob)))
        aucs_test.append(auc(fpr, tpr))

    plot_weight(acc_train, recall_train, f1_train, acc_test, recall_test, f1_test)

    print("Train accuracy: %.4f" % (sum(acc_train) / 10))
    print("Train recall: %.4f" % (sum(recall_train) / 10))
    print("Train F1: %.4f" % (sum(f1_train) / 10))
    print("Test accuracy: %.4f" % (sum(acc_test) / 10))
    print("Test recall: %.4f" % (sum(recall_test) / 10))
    print("Test F1: %.4f" % (sum(f1_test) / 10))
    print("Test AUC Score: %.4f" % (sum(aucs_test) / 10))
    return decision_tree


def decision_tree_show(decision_tree, name_list):
    class_temp = [str(i) for i in range(4)]
    dot_data = tree.export_graphviz(decision_tree, out_file=None,
                                    feature_names=name_list,
                                    class_names=class_temp,
```

```
                                    filled=True, rounded=True)

    print(type(dot_data))

    with open('./dot_data.dot', 'w', encoding='utf-8') as f:
        class_temp = [str(i) for i in range(18)]
        dot_data = tree.export_graphviz(decision_tree, out_file=f,
                                        feature_names=name_list,
                                        class_names=class_temp,
                                        filled=True, rounded=True)

    check_call([r'G:\py_env\Lib\site-packages\graphviz-2.38\release\bin\dot', '-v', '-Tpng', './dot_data.
dot', '-o', 'test.png'])


def RF(data_train, label_train, data_test, label_test):
    acc_train = []
    recall_train = []
    f1_train = []
    aucs_train = []
    acc_test = []
    recall_test = []
    f1_test = []
    aucs_test = []
    for iTrees in range(10, 310, 20):
    # for iTrees in range(250, 251):
        # iTrees = 100
        # depth = 50
        # 训练
        RFModel = ensemble.RandomForestClassifier(n_estimators=iTrees, min_impurity_split=0.1, n
_jobs=-1, oob_score=False, random_state=531)
        RFModel.fit(data_train, label_train)

        # Accumulate auc on test set
        train_predict = RFModel.predict(data_train)

        print("Train accuracy: %.4f" % accuracy_score(label_train, train_predict))
        acc_train.append(accuracy_score(label_train, train_predict))
        print("Train recall: %.4f" % recall_score(label_train, train_predict, average='weighted'))
        recall_train.append(recall_score(label_train, train_predict, average='weighted'))
        print("Train F1: %.4f" % f1_score(label_train, train_predict, average='weighted'))
        f1_train.append(f1_score(label_train, train_predict, average='weighted'))
        test_predict = RFModel.predict(data_test)
```

```
            print("Test accuracy: %.4f" % accuracy_score(label_test, test_predict))
            acc_test.append(accuracy_score(label_test, test_predict))
            print("Test recall: %.4f" % recall_score(label_test, test_predict, average='weighted'))
            recall_test.append(recall_score(label_test, test_predict, average='weighted'))
            print("Test F1: %.4f" % f1_score(label_test, test_predict, average='weighted'))
            f1_test.append(f1_score(label_test, test_predict, average='weighted'))
            confusion_mat = confusion_matrix(label_test, test_predict)

            test_predict_prob = RFModel.predict_proba(data_test)
            one_hot = []
            for i in label_test:
                temp = [0] * 4
                temp[i] = 1
                one_hot += temp
            fpr, tpr, _ = roc_curve(one_hot, list(itertools.chain.from_iterable(test_predict_prob)))
            print("Auc score: %.4f" %auc(fpr, tpr))

            # pd.DataFrame({'fpr': fpr, 'tpr': tpr}).to_csv(r'../results/classification/RF_data_enhance.csv',
            #                                               encoding='utf_8_sig', index=False)

            # print("Test confusion matrix:")
            # print(confusion_mat)
        plot_weight(acc_train, recall_train, f1_train, acc_test, recall_test, f1_test)


def XGB(data_train, label_train, data_test, label_test):
    acc_train = []
    recall_train = []
    f1_train = []
    aucs_train = []
    acc_test = []
    recall_test = []
    f1_test = []
    aucs_test = []
    # for estimator_num in range(50, 500, 20):
    for lr in np.arange(0.05, 0.8, 0.05):
    # for lr in np.arange(0.15, 0.16):
        estimator_num = 550
        m_class = xgb.XGBClassifier(learning_rate=lr, n_estimators=estimator_num, objective='multi:s
oftmax', num_class=4)
        # m_class = xgb.XGBClassifier(learning_rate=0.1, n_estimators=1000, max_depth=50, min_c
hild_weight=6, gamma=0,
```

```
        #                                       subsample=0.5, n_jobs=-1, reg_alpha=0.05, reg_lambda=
0.05,
        #                                       colsample_bytree=0.8, objective='multi:softmax', num_cl
ass=4, seed=127)
        # 训练
        decision_tree = m_class.fit(data_train, label_train)
        train_predict = m_class.predict(data_train)

        print("Train accuracy: %.4f" % accuracy_score(label_train, train_predict))
        acc_train.append(accuracy_score(label_train, train_predict))
        print("Train recall: %.4f" % recall_score(label_train, train_predict, average='weighted'))
        recall_train.append(recall_score(label_train, train_predict, average='weighted'))
        print("Train F1: %.4f" % f1_score(label_train, train_predict, average='weighted'))
        f1_train.append(f1_score(label_train, train_predict, average='weighted'))

        test_predict = m_class.predict(data_test)
        print("Test accuracy: %.4f" % accuracy_score(label_test, test_predict))
        acc_test.append(accuracy_score(label_test, test_predict))
        print("Test recall: %.4f" % recall_score(label_test, test_predict, average='weighted'))
        recall_test.append(recall_score(label_test, test_predict, average='weighted'))
        print("Test F1: %.4f" % f1_score(label_test, test_predict, average='weighted'))
        f1_test.append(f1_score(label_test, test_predict, average='weighted'))

        test_predict_prob = m_class.predict_proba(data_test)
        one_hot = []
        for i in label_test:
            temp = [0] * 4
            temp[i] = 1
            one_hot += temp
        fpr, tpr, _ = roc_curve(one_hot, list(itertools.chain.from_iterable(test_predict_prob)))
        print("Test auc score: %.4f" % auc(fpr, tpr))
        # pd.DataFrame({'fpr': fpr, 'tpr': tpr}).to_csv(r'../results/classification/XGB_data_enhance.csv',
        #                                               encoding='utf_8_sig', index=False)

        # confusion_mat = confusion_matrix(label_test, test_predict)
        # print("Test confusion matrix:")
        # print(confusion_mat)
    plot_weight(acc_train, recall_train, f1_train, acc_test, recall_test, f1_test)


def label_creat(list_level):
    labels = []
    for i in list_level:
```

```
            if i == "A":
                labels.append(0)
            elif i == "B":
                labels.append(1)
            elif i == "C":
                labels.append(2)
            elif i == "D":
                labels.append(3)
        return labels


def creat_level(lists):
        labels = []
        for i in lists:
            if i == 0:
                labels.append('A')
            elif i == 1:
                labels.append('B')
            elif i == 2:
                labels.append('C')
            elif i == 3:
                labels.append('D')
        return labels


def main():
        # path_data = r'../../../index.csv'
        # data_label = pd.read_csv(path_data, encoding='utf_8_sig')
        # data_split(data_label)
        train_data = pd.read_csv(r'../../../train.csv', encoding='utf_8_sig')
        test_data = pd.read_csv(r'../../../test.csv', encoding='utf_8_sig')
        index_temp = [
]
        train_sp = train_data[index_temp]
        # test_sp = test_data[index_temp]
        train_label = label_creat(list(train_data['信誉评级']))
        # test_label = label_creat(list(test_data['信誉评级']))
        # dt = DT(train_sp, train_label, test_sp, test_label)
        # decision_tree_show(dt, list(dict_name.keys()))
        # RF(train_sp, train_label, test_sp, test_label)
        # XGB(train_sp, train_label, test_sp, test_label)

        # train_data = pd.read_excel(r'../../problem  C  附件 1.xlsx')
```

```
    test_data = pd.read_csv(r'../../../index_2.csv', encoding='utf_8_sig')
    # train_sp = train_data[[i for i in range(1, 21)]]
    test_sp = test_data[index_temp]
    m_class = xgb.XGBClassifier(learning_rate=0.15, n_estimators=350, objective='multi:softmax', nu
m_class=4)
    m_class.fit(train_sp, train_label)
    test_predict = m_class.predict(test_sp)
    train_predict = m_class.predict(train_sp)
    print("Train F1: %.4f" % f1_score(train_label, train_predict, average='weighted'))
    test_data['信誉评级'] = creat_level(test_predict)
    test_data.to_csv(r'../../../附件二企业信誉评级识别结果.csv', encoding='utf_8_sig', index=False)


if __name__ == '__main__':
    main()
```

附录二：支撑材料文件列表

（1）data preprocess.py

（2）predict.py

（3）附件二企业信誉评级识别结果.xlsx