考虑突发因素的中小企业信贷策略建模研究

摘要

实际情况中，由于小微企业的经营规模和抵押资产有限，银行主要依据其历史信贷信息和经营状况评价企业的借贷风险，同时指定相应的借贷策略。对此，本文建立数学模型量化企业的借贷风险，并设计各类型企业的借贷方案，研究在突发因素影响下企业的借贷放案调整策略，本文的具体工作如下：

（1）基于企业的信用等级、违约记录和进销发票信息建立借贷风险评估指标体系，包括企业盈利及运营能力指标、企业交易结构及成长类型指标、供应链结构特征指标和企业信用指标四大类，15 小类指标体系。然后，选则基于熵值取权的层次分析法对上述指标确定权值，获得企业的综合风险评估打分。最后，基于企业的信誉等级和经营状况，设计不同信誉等级企业的借贷金额和借贷年利率范围约束，确定各企业的借贷策略。

（2）考虑到附件二所给企业信息中缺少信誉等级和是否违约数据，对此，本文首先利用主成分分析法对上述企业风险评估指标体系进行降维处理，获得主成分。然后利用深度神经网络模型实现企业信誉等级的自动识别，实验结果分析知，深度神经网络模型在训练集上的识别准确率为 100%。同时，基于附件三所给数据，考虑建立年贷利率与用户流失率的函数关系，对问题一中的借贷方案进行修正。对于信誉评级为 D 的用户考虑不给予借贷，而对于信誉评级为 C、B、A 的企业，其借贷金额集中在 30,50 和 70 的附近。

（3）考虑实际情况中不同的突发因素类型和不同的企业类型，其经营状况所受影响的性质（积极或消极）和程度将有所区别。对此本文从三方面考虑突发因素对于企业借贷的影响，包括：突发因素对企业的影响程度、突发因素的严重程度以及突发因素对企业的性质，其中突发因素的严重程度由影响范围和作用时间确定。基于上述三个因素建立银行借贷金额修正模型。同时以新冠疫情为例分析企业借贷方案的调整策略。附件二所给企业中存在 9 家企业与医药、生物相关，本文假设其受“积极”影响，而其它行业为消极“影响”。结果表明对于信誉等级高且规模大的企业其贷款金额在原有的基础上呈上浮趋势而对于信誉等级低且规模小的企业，贷款金额则呈下降趋势。同时，企业贷款年利率均在原有基础上有小幅下降。

关键词：中小企业，风险等级评价体系，信誉评级自动识别，信贷方案，突发因素

一、问题重述

对于中小企业，由于其企业规模、抵押资产有限，因此银行一般考虑基于其历史信誉记录如是否存在违约记录、信誉评级，以及企业的经营状态信息，如进、销发票数据等设计信贷方案，在满足企业需求的同时实现自身收益的最大化和风险的最小化。一般信贷方案包括：贷款金额、贷款年限、贷款利率等。对此，本文建立数学模型对如下问题进行研究：

（1）基于企业的基本信息（企业名称、信誉等级、是否违约）进销发票数据对企业信贷风险进行量化分析，同时给出在信贷总额固定时的各个企业信贷策略；

（2）基于问题一研究结果，分析在企业信誉等级和是否违约数据缺失情况下，给定信贷总额的各个企业信贷策略；

（3）考虑各类突发因素，如政策的调整、重大灾难等对不同类型、不同行业企业经营的影响，基于问题而研究结果，给出在突发因素的影响下，各企业信贷方案的调整策略。

二、模型假设

（1）假设附加一附件二中数据分布相同；

（2）假设附件数据真实可靠；

（3）设突发因素将对所有企业的经营状况带来一定的影响，该影响可为积极或消极，可大可小；

（4）假设新冠疫情对娱乐、餐饮等行业带来负面影响，而对医药、生物等行业带来正面影响；

三、符号变量说明

符号	符号含义
$P(L=l,B=b)$	企业信誉等级为l是否违约记录b的联合概率分布
t	发票平均税率
GN	销方基尼系数
$HINI$	销方 HIHI 指数
C	银行总借贷额
x_i	贷款额度
λ_i	贷款利率
m_{out},m_{in}	销项发票总额（含税）、进项发票总额（含税）
$P(y=j\mid x)$	企业信誉等级概率分布
x_{rate}	借贷利率
y_{loss}	客户流失率
$x^{'}$	修正后的贷款金额
v	突发因素对企业的性质
d	突发因素的影响程度
$m(x)$	突发因素对企业的影响程度

四、问题分析

4.1 问题一分析

问题一要求对所给企业信贷风险进行量化分析，同时给出当银行年度信贷总额固定时的信贷策略。对此，本文首先基于条件概率和联合分布概率分析企业信誉评级和违约记录间的关联关系。然后采用层次分析法，考虑企业盈利及运营能力、企业交易结构及成长类型、企业供应链结构和应用设计 15 个二级指标包括销售规模、税价比、企业基尼系数等，对企业信誉等级和违约风险进行建模。同时利用熵值取权法对权值进行修正，获得企业的量化风险。最后基于企业的信誉评级和运营情况对信贷策略进行建模。

4.1 问题二分析

问题二要求在问题一的基础上，分析当银行年度信贷总额为给定值时各企业具体信贷策略。由于附件二所给企业数据缺失信誉等级信息。对此，本文首先利用主成分分析法对问题一建立的指标体系进行降维处理获得主成分。然后，设计深度神经网络模型，根据附件一数据训练模型实现企业信誉等级的自动识别。同时，结合附件三数据建立年贷利率与客户流失率的数学模型，并基于此对问题一模型进行修正，获得企业的信贷策略。

4.3 问题三分析

问题三要求考虑突发因素对企业经营状况的影响，同时企业信贷方案的调整策略。对此本文认为不同的突发因素对不同企业的影响不同，考虑突发因素的严重程度、突发因素对企业的性质（积极或消极）、突发因素对企业的影响程度这三个方面在问题二的基础上设计贷款金额修正模型。同时以 Conv-19 为例分析附件二所给企业贷款策略的调整结果。

五、模型的建立与求解

5.1 信贷风险量化分析与信贷策略建模研究

中小微企业的信贷风险量化分析与管理是银行选择是否放贷与贷款策略制定的重要依据。实际情况中，信贷风险分类是风险评价的重要方法。对此本文首先依据附件一所给企业信誉评级和历史是否违约记录，建立企业基本信贷风险量化模型。同时在此基础上还考虑融入企业发票信息，进行综合评价。最后，在上述基础上建立信贷策略模型，考虑借贷金额、借贷年利率和借贷概率，使得银行获利最大。

5.1.1 基于层次分析法的借贷风险量化建模

信用评级是量化借贷风险的重要指标之一，本文首先基于信用评级和用户历史违约记录建立基本信贷风险分析模型，然后对基于企业的发票及纳税信息分析企业的运营情况，建立层次指标体系确定最后企业借贷风险。

（1）企业基本信贷风险量化模型

附表 1 中所给 120 余家企业的信誉评级和是否违约数据统计分析结果如下图所示：

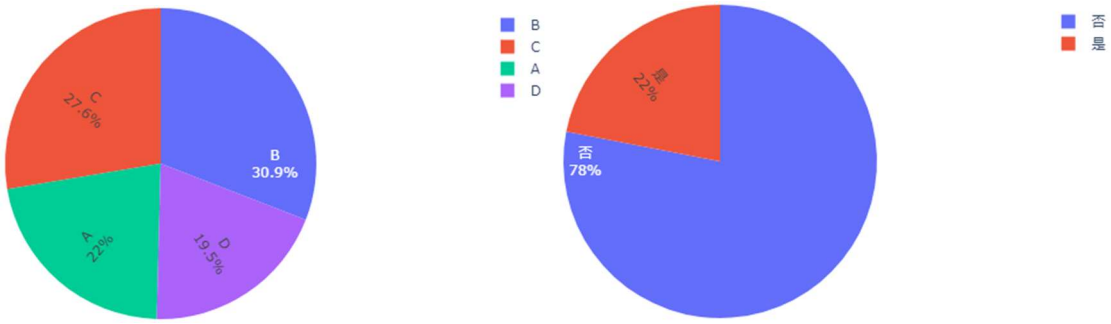

图 5.1 企业信誉评级和是否违约记录分布饼图

如图 5.1 所示，企业信誉评级分布相差不大其中信誉评级为 B 的企业占比最高维 30.9%，信誉评级为 D 的企业占比较低为 19.5%。绝大部分（约 78%）的企业均不存在违约记录，仅有 22%的企业存在违约。本文基于联合概率和条件概率分析企业信誉评级和是否违约记录的依存情况，如下式所示：

$$\begin{cases} P(L=l,B=b)=\dfrac{\#(L=l)\cap\#(B=b)}{\#(L=l)\cup\#(B=b)} \\ P(L=l\mid B=b)=\dfrac{P(L=l,B=b)}{P(B=b)} \\ P(B=b)=\dfrac{\#(B=b)}{\#(B=\bullet)} \end{cases} \quad ，l\in\{a,b,c,d\},b\in\{1,0\} \qquad (1)$$

如式（1）所示，$P(L=l,B=b)$ 表示某一企业信誉等级为 $l(l\in\{a,b,c,d\})$，是否违约记录（$b\in\{1,0\}$，1 表示存在违约记录，0 表示无违约记录）的联合概率分布，若该值越大则表示两变量相关程度越高。$P(L=l\mid B=b)$ 表示在企业是否违约记录为 b 的条件下，企业信誉评级为 l 的条件概率。同理若该值越大则表示 b 对 l 的单向依存关系越强。统计不同信誉等级、不同违约记录下的联合概率分布，绘制热力图如下：

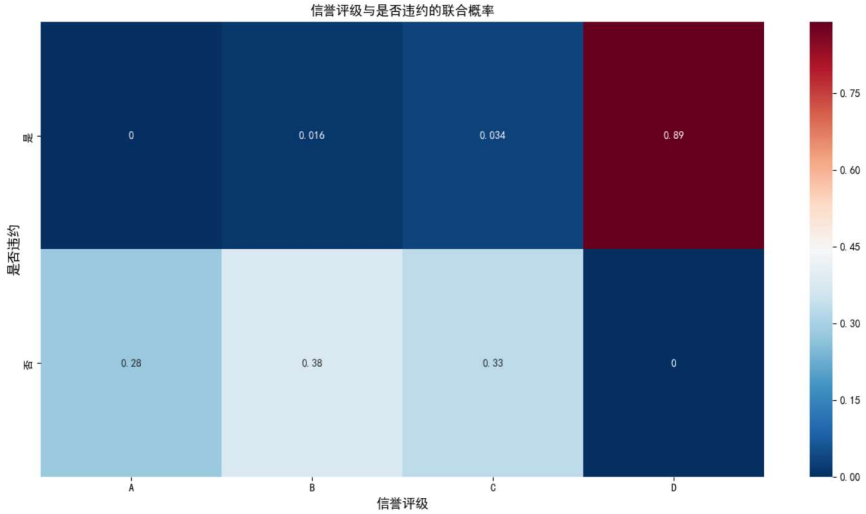

图 5.2 企业信誉评级和是否违约记录联合概率分布热力图

观察 5.2 热力图可以看出各信誉等价和是否违约的联合概率大小。在基于附件一的有限数据下可以得到如下结论：

① 信誉评级为 A 的用户存在违约记录的可能性（概率）为 0；

② 信誉评级为 D 的用户存在违约记录的可能性（概率）为 1；

③ 不存在违约的用户其信誉评级在 A，B，C 上的分布较为平均；

④ 存在违约的用户极大可能信誉评级为 D；

因此，可以判断若用户存在违约行为其信誉评级即有可能为 D，即最低等级。参考相关文献的企业评级和风险度量表[1]如下：

表 5.1 中小微企业信评级和风险度量表

信誉评级	风险等级	风险描述
AAA	低风险（极低）	特大型或大型优质客户如大型金融机构或银行，在国内具有较强竞争优势，经营管理规范，信用状况极其稳定，发展前景很好，具有极强的抗打击能力和可靠性
AAA-	低风险（极低）	特大或大型企业，在国内行业内具有领先的竞争优势，经营管理规范，信用状况非常稳定，经营实力和财务实力雄厚，发展前景很好，具有极强的抗打击能力和可靠性
AA+	低风险（很低）	企业有一定规模，在国内行业内具有明显的竞争优势，经营管理规范，信用状况很稳定，具有很强的财务实力，发展前景较好，有很强的抗打击能力和可靠性
AA	低风险（较低）	在国内行业内部企业具有很强的竞争优势，经营管理很规范，信用状况较稳定，具有很强的财务实力，具有一定的抗打击能力和可靠性
AA-	低风险（略低）	在国内行业内部企业具有较强竞争优势，经营管理规范，信用状况较稳定，具有较强的财务实力，能够承受一定的打击
A+	低风险	在国内行业内部企业具有一定的竞争优势，经营管理规范，信用状况稳定，具有一定的财务实力，拥有一般的抗打击能力且未来一段时间内不太可能出现显著影响企业履约意愿和偿债能力的突发情况
A	中风险（较低）	在国内行业内企业具有一定的竞争优势，经营管理规范，信用状况稳定，具有一定的财务实力，拥有一定的抗打击能力，但潜在的内外部不利因素会在某种程度上影响企业履约意愿和偿债能力，但一般情况下短期内影响不严重
A-	中风险（略低）	企业业绩表现较好，经营管理较规范，在正常经营情况下，企业能够为信贷业务提供充足的财务保障，信用状况一般，但潜在的内外部不利因素会在某种程度上影响企业偿债能力和履约意愿，但一般情况下短期内影响不会很严重
BBB+	中风险	企业业绩表现良好，经营管理较规范，企业能够为信贷业务提供较好的财务保障，信用状况一般，但企业在未来一段时间内有可能面对一些影响其偿债能力和履约意愿的不利因素
BBB	中风险（较高）	企业业绩表现和经营管理状况一般，企业虽能够为信贷业务提供相应的财务保障且信用状况尚可，但企业在短期内有可能面对一些影响其履约意愿和偿债能力的不利因素
BBB-	中风险（高）	企业业绩表现和经营管理状况一般，企业虽能够为信贷业务提供一定的财务保障且信用状尚可，但内外部不利因素将可能为企业信用状况带来一定影响
BB+	高风险	企业业绩表现及管理水平一般，信用状况存在一定的波动和不确定性，一旦出现不利因素企业信用状况将迅速恶化
BB	高风险（较高）	企业业绩表现不佳，管理水平较差，生产经营面临很大压力且逐步恶化，可持续经营能力无法保证，信用状况较弱
B	高风险（很高）	企业业绩表现持续下滑，管理水平很差，生产经营状况持续恶化，可持续经营能力存在极大问题，信用状况极弱
CCC	高风险（濒临违约）	企业管理水平极差，生产经营状况极为艰难，几乎无法维持经营，还款压力极高，还款意愿极地，虽未出现违约行为，

		但未来违约可能性极高
D	违约	发生违约行为，企业已基本或完全结束经营活动，企业已不拥有任何资产或仅拥有极少量资产，银行债权基本损失

如表 5.1 所示，其将企业信誉评级和风险等级划分为 12 级，其中信誉等级 A 表示企业基本无借贷风险，信誉等级为 D 企业表示其已发生违约行为，银行应拒绝为该类客户提供借贷服务。信誉等级为 B 和 C 的客户银行可以考虑该企业经营管理的实际情况制定不同的借贷策略。

上述企业风险评级策略较为粗略，对此本文还考虑分析附件中的各企业发票和报税信息，量化企业的经验管理风险。同时，考虑建立企业信贷风险综合评价指标体系，分析企业借贷风险。

（2）考虑层次分析法的企业基本信贷风险量化模型

基于层次分析法本文首先考虑四个一级指标包括企业盈利及运营能力指标、企业交易结构及成长类型指标、供应链结构特征指标、企业信用指标。各个指标介绍如下：

① 企业盈利及运营能力指标：主要由企业的收入及成本两个角度即销项发票和进项发票对企业经营状况进行量化。可包括销售规模、销项开发数量、平均销项开票金额、进购规模、进项开发数量、税价比及发票平均税率等；

② 企业交易结构及成长型指标：考虑供应链结结构，即衡量企业与供需方在一段时间内的适应性及共生能力。因此这里主要考虑企业与下游企业的交易结构及成长性，定义下游客户数量、指定周期冲正发票金额比率、指定周期作废发票金额比率、指定周期下游客户数 4 个指标；

③ 供应链结构特征指标：考虑企业与下游厂商的结构是否稳定、合理，设计销方基尼系数和销方 HIHI 指数；

④ 企业信用指标：直接反映企业累积信誉情况以及违约风险大小，包括企业信誉评级、违约记录；

基于上述介绍，设计企业信贷风险评价体系，如表 5.2 所示：

表 5.2 企业信贷风险评价体系

一级指标	二级指标	描述	类型
企业盈利及运营能力	销售规模	销项发票总金额之对数	收益型
	销项开票数量	销项发票总数量之对数	收益型
	平均销项开票金额	销售规模与销项开票数量之比（不取对数）	收益型
	进购规模	进项发票总金额之对数	收益型
	进项开发数量	进项发票总数量之对数	收益型
	税价比	指定周期内（近 12 个月）销项发票总税额和总税价额之比	收益型
	发票平均税率	企业在观测期内销项发票票面增值税额与票面销售额之比	收益型
交易结构及成长类型	下游客户数量	所有发生交易行为的客户总数量	收益型
	指定周期负数发票金额比率	指定周期（近 12 月）销项负数票总金额与总金额比率	成本型
	指定周期作废发票金额比率	指定周期（近 12 月）销项作废发票总金额与总金额比率	成本型
	指定周期下游客户数	指定周期（近 6 月）发生交易的客户数	收益型
供应链结构	销方基尼系数	销方基尼系数是衡量企业交易结构集中度的指标	成本型

	销方 HIHI 指数	销方 HHI 系数是另一个衡量下游客户集中度的指标，由每家下游企业交易额占比的平方和来表示	成本型
企业信用指标	企业信誉评级	包括 A,B,C,D 是个等级，其中 A 级表示信誉最后，D 级信誉最差	收益型
	违约记录	二值变量反映企业是否存在历史违约记录	成本型

如表 5.2 所示，企业信贷风险体系共包含四大类一级指标，15 小类二级指标。其中指标类型分为收益型和成本型，即收益型指标数值越大企业评价越好。反之成本型指标数值越大，其评价越差。其中，一级指标均为收益型指标，对于成本型指标在计算打分时一般需进行一定处理将其转化为收益型指标。其中关键指标的解释包括：

①销售规模

销售规模由求和评估企业在指定周期内销项发票的票面金额（不含税），然后计算对数值获得。由于实际情况中会出现起征点及减免税等因素的影响，因此应税销售额一般小于企业的实际销售额。销售规模是衡量企业规模的主要指标之一，其可间接反映企业的信用风险评估。

②开票数量

开票数量为评估企业在指定周期内销项发票开票数量的对数值。若开票限额确定，则企业的开票数量将与企业的业务量相关。

③发票平均税率

发票平均税率定义为评估企业在指定周期内销项发票税额与票额之比。其反映企业所销售的产品的增值税综合税率。在确定发票平均税率后，一般需对其进行进一步评估分档，如表 5.3 所示：

表 5.3 发票平均税率得分表

销项发票平均税率 t	得分
t≥15%	5
10%≤t<15%	3
6%≤t<10%	1
t<6%	0

④销方基尼系数

参考社会科学中对基尼系数的定义，这里定义销方基尼系数反映企业对销售额的集中程度，也即企业交易结构的击中程度，如下：

$$GN = 1 - \frac{1}{n}(2\sum_{i=1}^{n-1} CM_i + 1) \tag{2}$$

式（2）中，n 为该企业的购方数目，CM_i 为剔除最高交易金额所对应的购方后，购方金额由高至低的累积占比。若 GN 值越大则表明该企业的客户结构越集中。

⑤销方 HIHI 指数

HIHI 指数即为赫芬达尔—赫希曼指数，其由每家交易企业交易额占比的平方和表示。HHI 值越大则表示客户集中度越高。其计算公式如式（3）所示：

$$HHI = \sum_{i=1}^{m} a_i^2 \tag{3}$$

上式中，a_i即为销售额第$top\ i$的企业销售额占比。实际情况中m一般取值为 50。

⑥客户数

客户数是指定周期，（6 个月）发生交易客户的数量。对比开票数量，客户数的提升表明企业业务发展状态良好，同时客户数目的持续上升则显示该企业市场占有率的提升。

这里对上述指标进行统计分析，绘制销售规模、销项开票数量、平均销项开票金额等指标的箱型图如下所示：

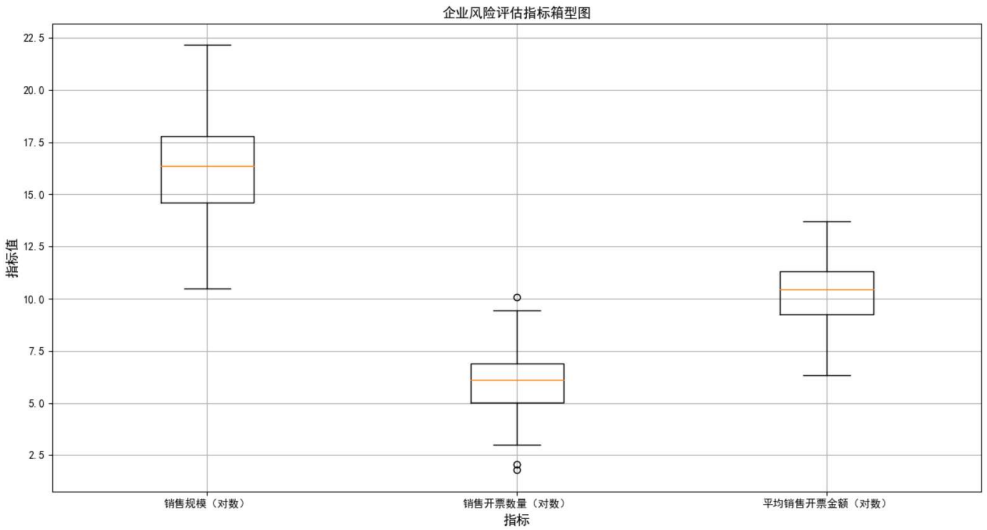

图 5.3 销项发票相关指标箱型图

如图 5.3 所示，这里对销项发票相关指标绘制箱型图，可以看到在 120 余家企业中其销售规模、销售开票数量、平均开票金额的分布大致呈正态分布。但对于开票数量部分企业该指标在统计学分析中表现为离群点。各企业的风险量化评价指标计算结果如下所示：

企业代号	销售规模（对数）	销项开票数量（对数）	平均销项开票金额（对数）	进购规模（对数）	进项开票数量（对数）	税价比	发票平均税率	下游客户数量	指定周期负数发票金额比率	指定周期作废发票金额比率	固定周期下游客户数量	销方基尼系数	销方HIHI指数
E1	22.1471	9.00085	13.14624	5.64922	0	0.135	0.15551	360	0.03131	0	360	0.99963	0.00000
E2	19.72651	6.34212	13.38439	4.9049	0	0.031	0.03149	27	0.00282	0	27	0.99472	0.00035
E3	13.44556	4.59512	8.85044	5.31586	0	0.029	0.03	13	0.18051	0	13	0.9697	0.04170
E4	13.35171	4.14313	9.20857	5.37082	0	0.134	0.15436	2	0	0	2	0.95238	0.02187
E5	12.90726	5.17615	7.73111	6.23388	0	0.133	0.15322	105	0.06763	0	105	0.98305	0.01835
E6	14.47785	4.15888	10.31897	4.53174	0	0.07	0.07526	20	0.00689	0	20	0.95312	0.03514
E7	12.5499	3.04452	9.50538	5.57652	0	0.029	0.03	10	0.20721	0	10	0.85714	0.20165
E8	13.748	4.80402	8.94398	5.57652	0	0.029	0.03021	101	0.02806	0	101	0.97541	0.00559
E9	13.36908	5.03044	8.33864	5.16341	0	0.029	0.03015	105	0.03428	0	105	0.98039	0.00932
E10	13.69483	3.98898	9.70585	5.57652	0	0.057	0.06	4	0	0	4	0.94444	0.07192
E11	12.16249	3.49651	8.66598	5.21792	0	0.029	0.03	19	1.00E-05	0	19	0.90909	0.04147
E12	13.40365	3.49651	9.90714	6.60639	0	0.03	0.03114	17	0	0	17	0.90909	0.07961
E13	18.84264	7.0184	11.82424	3.49624	0	0.087	0.095	44	0.0006	0	44	0.99731	0.00185
E14	12.49644	4.41884	8.0776	5.57652	0	0.029	0.03	37	0	0	37	0.96386	0.02746
E15	14.15839	5.52146	8.63693	6.65906	0	0.155	0.18353	11	1.25186	0	11	0.988	0.08675
E16	14.17684	3.09104	11.0858	7.15012	0	0.145	0.16937	2	0.05917	0	2	0.86364	0.06176
E17	12.59873	3.85015	8.73858	5.57652	0	0.139	0.16186	4	0	0	4	0.93617	0.03565
E18	12.44443	4.31749	8.12694	4.48289	0	0.029	0.03006	23	0.01807	0	23	0.96	0.02099
E19	11.56218	1.79176	9.77042	5.57652	0	0.029	0.03	4	0	0	4	0.5	0.24891
E20	12.45811	3.85015	8.60796	5.15225	0	0.029	0.03002	25	0	0	25	0.93617	0.02992
E21	13.48247	3.78419	9.69828	7.34155	0	0.003	0.00307	1	0	0	1	0.93182	0.04020
E22	12.65511	4.96981	7.6853	5.57652	0	0.029	0.03004	118	0.0016	0	118	0.97917	0.07276
E23	10.49563	3.04452	7.45111	4.36641	0	0.029	0.03	17	0	0	17	0.85714	0.05941
E24	19.30732	5.65249	13.65483	4.483	0	0.079	0.08607	5	0.00379	0	5	0.98947	0.00080
E25	12.15832	3.3673	8.79102	5.29832	0	0.029	0.03	26	0	0	26	0.89655	0.25477
E26	11.55834	5.22575	6.33259	5.29832	0	0.112	0.12563	123	0	0	123	0.98387	0.00476

图 5.4 各企业风险评价指标计算结果（截取部分）

上图 5.4 为各企业风险评价指标计算结果，完成结果见附件。

（3）基于熵值取权的层次分析法权值确定

目前，基于指标体系的评价方法主要可分为两大类，第一类可视为主观赋权法，即利用综合咨询评分的方法确定权重，该类方法一般包括模糊综合评价法、层次分析法、功效系数法以及综合指数法等。而另一类为客观赋权法，即根据指标间的相关关系或各指标值的变异程度确定权数，其代表模型为主成分分析法、因子分析法、TOPSIS 法又称为理想值法等。本文考虑基于层次分析法（AHP）对上述企业风险等级指标体系进行建模，量化各企业的信贷风险，同时考虑到主观赋权的偏差，这里使用结合熵值取权法对权值进行修正。

a. 层次分析法

在对社会、经济以及相关工程或科学管理领域及系统工程领域等问题的分析建模中，经常会面临一个由众多相互关联、制约因素构成的复杂系统。对于该类问题的分析和建模，层次分析法[2]得到了广泛的应用。

层次分析法（Analytic Hierarchy Process，AHP）于上世纪 70 年代由美国运筹学家 Saaty 教授提出，其对于复杂模糊的决策问题的解决提供了一种有效的手段，尤其适用于难于完全由定量分析方法解决的问题。层次分析法建模包括如下四个步骤：

Step1. 结构问题，确定指标，建立层次结构模型；

Step2. 确定各层次中所有指标的判别矩阵；

Step3. 层次单排序一致性检验；

Step4. 层次总排序一致性检验；

考虑到层次结构虽整体上反映了各因素间的关联关系，但由于各指标权值的确定仍需依据专家打分矩阵确定，该方法存在较强的主观性。由于其说服力及可信度较差，因此在实际情况中一般需要通过其它方法对权值进行修正，本文考虑熵值法[3][4]。

b. 熵值取权法

信息论中一般选择熵衡量事件发生的不确定性程度，其衡量信息的信息量的大小。若信息量越大，则表明该事件的不确定性越小，也即熵也就越小。反之，若信息量越小，则不确定性将越大，故熵也越大。故熵值的大小可以较好反映某个值标的离散程度，同时指标的离散程度越大，则表明该指标对综合评价的影响越大，也即“信息量”越大。因此，基于信息熵表示各项指标的变异程度，进而确定不同指标权值，已修正 AHP 层次分析法的“主观性”影响。

综上可知，熵值取权法可视为一种客观赋权法，其根据各项指标所提供信息量的大小确定权重。记有 m 个方案，n 项评价指标，则原始指标矩阵为 $X=(\mathrm{x}_{ij})_{m\times n}$，对于某项指标 x_j，若 X_{ij} 的差距越大，则表明该指标在综合评价中所起的作用将越大。若某一指标全部相等，则表明该指标在综合评价中不提供任何参考以及或“线索”。

熵值取权法的具体步骤如下：

Step1. 数据预处理，包括指标的一致化（统一为“收益性”指标）和无量纲化处理；

Step2. 计算方案 i 在指标 j 中所占权值：

$$P_{ij}=\frac{X_{ij}}{\sum_{i=1}^{n}X_{ij}},\left(j=1,2,...,m\right)$$

上式中，X_{ij} 为指标 j 下方案 i 的取值；

P_{ij} 为指标 j 下方案 i 占该指标的权值；

Step3. 确定指标 j 熵值：

$$e_j=-k\bullet\sum_{i=1}^{n}P_{ij}\log\left(P_{ij}\right)$$

$$k=\frac{1}{\ln\left(n\right)}>0$$

上式中，e_j 即为指标 j 的熵值；

Step4. 计算指标 j 的变异系数

对于指标 j，指标值 X_{ij} 的差异越大，则表明其对方案评价的贡献越大，熵值将越小：

$$g_j = 1 - e_j$$

Step5. 权值确定：

$$\omega = \frac{g_j}{\sum_{j=1}^{m} g_j}, j = 1,2,...,m$$

由熵值取权法修正后 AHP 各指标的权值如下所示：

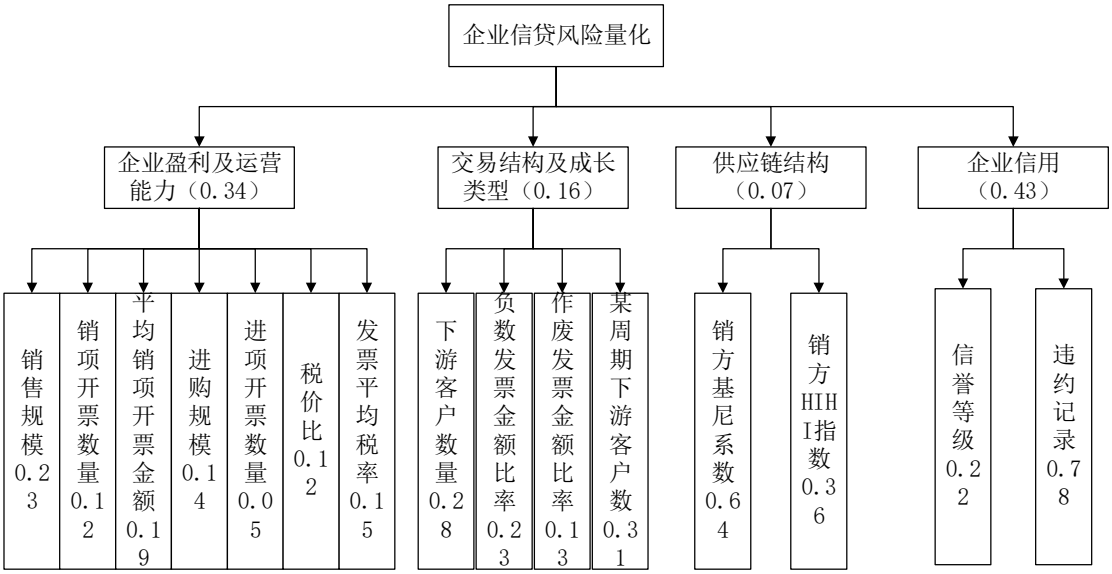

图 5.5 企业信贷风险量化指标体系及权值

首先将上述指标体系相关指标转化为“收益性”指标，然后计算各企业各指标的取值，依据该取值及该指标的分布确定该指标的打分，并基于层次分析法分计算综合得分。最后将该得分映射至表 5.1 所对应的 12 等级中确定该企业的风险等级。如下表所示：

表 5.4 各企业风险评估评价指标打分即综合得分（截取部分）

企业代号	销售规模（对数）	销项开票数量（对数）	平均销项开票金额（对数）	进购规模（对数）	进项开票数量（对数）	税价比	发票平均税率	下游客户数量	指定周期负数发票金额比率	指定周期作废发票金额比率	固定周期下游客户数量	销方基尼系数	销方HHI指数	信誉评级	是否违约	综合得分
E1	5	5	5	3	5	5	5	5	1	1	5	1	5	5	5	4.462
E2	5	3	5	1	5	2	2	3	4	1	3	3	4	5	5	3.994
E3	1	1	1	2	5	2	2	2	1	1	2	5	1	2	5	2.912
E4	1	1	2	2	5	5	5	1	4	1	1	5	1	2	5	3.269
E5	1	2	1	4	5	5	5	4	1	1	4	4	2	4	5	3.685
E6	2	1	3	1	5	3	3	2	3	1	2	5	1	5	5	3.523
E7	1	1	2	3	5	2	2	1	1	1	1	5	1	5	5	3.215
E8	1	2	1	3	5	2	2	4	1	1	4	4	2	5	5	3.456
E9	1	2	1	2	5	2	2	4	1	1	4	4	2	5	5	3.408
E10	1	1	2	3	5	3	3	1	4	1	1	5	1	4	5	3.323
E11	1	1	1	2	5	2	2	2	4	1	2	5	1	2	5	3.023
E12	1	1	3	4	5	2	2	2	4	1	2	5	1	4	5	3.439
E13	5	4	5	1	5	3	3	3	4	1	3	2	3	5	5	4.057
E14	1	1	1	3	5	2	2	3	4	1	3	5	1	2	5	3.166
E15	1	2	1	4	5	5	5	1	1	1	1	4	1	5	5	3.47
E16	1	1	4	4	5	5	5	1	1	1	1	5	1	5	5	3.669
E17	1	1	1	3	5	5	5	1	4	1	1	5	1	5	5	3.537
E18	1	1	1	1	5	2	2	2	2	1	2	5	2	5	5	3.211
E19	1	1	2	3	5	2	2	1	4	1	1	5	1	5	5	3.326
E20	1	1	1	2	5	2	2	2	4	1	2	5	1	4	5	3.213
E21	1	1	2	4	5	1	1	1	4	1	1	5	1	4	5	3.187
E22	1	2	1	3	5	2	2	4	4	1	4	4	1	5	5	3.542
E23	1	1	1	1	5	2	2	2	4	1	2	5	1	4	5	3.165
E24	5	3	5	1	5	3	3	1	4	1	1	3	3	5	5	3.871
E25	1	1	1	2	5	2	2	3	4	1	3	5	1	2	5	3.118
E26	1	2	1	2	5	4	4	4	4	1	4	4	2	5	5	3.703
E27	1	2	1	3	5	2	2	4	2	1	4	4	1	5	5	3.468
E28	1	1	1	3	5	4	4	1	4	1	1	5	1	4	5	3.35
E29	5	5	3	3	5	4	4	5	2	1	5	1	3	2	1	2.602
E30	5	5	4	4	5	5	5	5	3	1	5	1	3	4	5	4.374
E31	5	5	5	2	5	2	2	1	4	1	1	1	5	5	5	3.869
E32	5	3	5	2	5	1	1	3	4	1	3	3	3	4	5	3.83
E33	5	4	5	1	5	3	3	5	2	1	5	2	3	4	5	4.078
E34	5	3	5	1	5	2	2	3	3	1	3	3	3	4	5	3.837
E35	5	5	4	4	5	5	5	5	2	1	5	1	5	4	5	4.387

如表 5.4 所示，若企业综合得分越高则表明该企业的信贷风险越低，这里每项指标及综合得分均按 5 分制评价。根据表 5.4 相关结果绘制不同企业综合得分的分布如下：

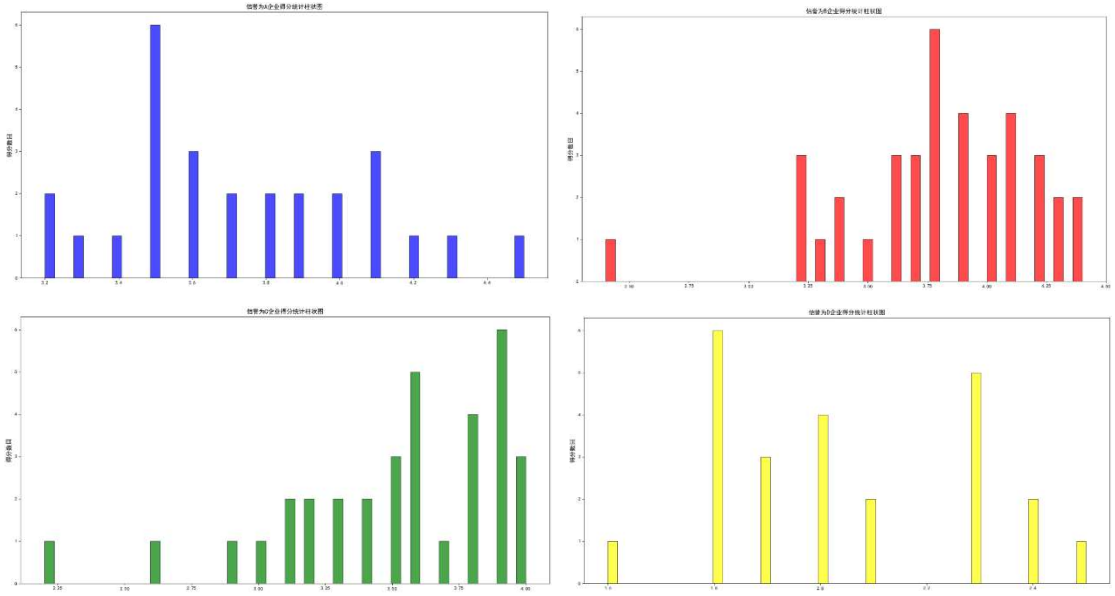

图 5.6 不同信誉等级下企业综合得分柱状图

如图 5.6 所示，从左至右，从上至下分别表示信誉等级 A、B、C、D 企业的综合得分分布，可以看出信誉等级为 A 的企业综合得分均分布至 4.0 附件，而信誉等级为 D 的企业得分均集中在 2.0 附近。

5.1.2 考虑企业风险等级和经营规模的借贷策略

商业银行对于企业的借贷策略一般需参考客户的风险等级、经营规模等指标评估借贷风险和收益，综合确定借贷策略。本文首先基于上述风险等级评价指标体系确定的企业分线等级完成客户分类，同时针对不同风险等级的用户确定其借贷金额区间，以及不同规模下的利率区间。考虑到题目指定借贷周期为 1 年，因此本文不对借贷时间进行讨论。如下：

表 5.5 不同客户类型对应借贷规模及利率区间

企业分类	信誉评级	风险等级	借贷规模（万元）	借贷利率%
发展类	A	低	100	4%
			80≤x<10	4.5%
			50≤x<80	4.5%~6%
			30≤x<50	6%~7%
			20≤x<30	7%~8%
			10≤x<20	8%~10%
巩固类	B	中	50≤x<80	5%~6%
			30≤x<50	6%~8%
			20≤x<30	8%~10%
			10≤x<20	10%~12%
调整类	C	高	20≤x<50	8%~10%
			10≤x<20	10%~15%
淘汰类	D	极高	10≤x<20	15%

如表 5.5 所示，首先基于信誉评级和风险等级将企业分为四大类，同时针对不同风险等级设置不同借贷规模（最高为 1000 万以上，最低少于 50 万）并针对

不同的借贷规模确定不同的借贷利率，根据题目背景最高为15%，最低为4%。基于表5.5同时结合企业经营状况，即销项总开票金额与进项总开票金额直差确定最终的借贷策略。

基于题目要求，记银行总借贷额为C，共有n家企业要求借贷，且每家企业的银行给定的贷款额度为x_i，贷款利率为λ_i，贷款期限为1年。则银行总收益P如下所示：

$$P=\sum_{i=1}^{n}(1+\lambda_i)x_i$$

$$s.t.\begin{cases}\sum_{i=1}^{n}x_i=C\\ \mu_{up}\le\dfrac{x_i}{|m_{out}-m_{in}|}<\mu_{down}\\ x_{down}\le x<x_{up}\\ \lambda_{up}\le x<\lambda_{down}\end{cases}\qquad(2)$$

如式（2）所示，其中$x_{up},x_{dowm},\lambda_{up},\lambda_{down}$分别表示银行批准的贷款额度及利率的上下边界，其由企业风险等级同时结合表5.4共同确定。m_{out},m_{in}表示企业的销项发票总金额（含税）与进项发票总金额（含税），基于$|m_{out}-m_{in}|$即可简单反映企业的经营状态。μ为超参数，通过调整该超参数即可控制银行的借贷收益和风险。

5.2 固定贷款金额下的银行信贷策略建模

由5.1小结式（2）分析知，对于信贷策略的设计需首先确定企业的信誉评级，因此本文首先利用组成分分析法对指标体系进行指标筛选，然后设计深度神经网络对附件二中的企业信誉等级进行分类。最后利用分类结果确定各企业的贷款金额和贷款利率。

5.2.1 基于主成分分析法的企业风险评估指标的筛选

主成分分析[5]本质上是一种数据降维的方法，其主要将高维数据通过协方差矩阵映射至低维空间中，使得映射后的数据极大的保留其原始特征。因此PCA也可以视为是一种特征挖掘和表示的有效手段。PCA的主要步骤如下：

Step1. 计算每一指标数据的平均值$u[m]=\dfrac{1}{N}\sum_{n=1}^{N}X[m,n]$；

Step2. 对每一指标数据中各个元素减去平均值，消除误差$B=X-uh$；

Step3. 计算协方差矩阵：

$$C=E[B\otimes B]=E[B\bullet B^*]=\frac{1}{N-1}\sum B\bullet B^*$$

其中，B^*即为矩阵B的共轭转置；

Step4. 计算矩阵C的特征值和特征向量，$V^{-1}CV=D$，其可由矩阵分解求解实现；

通过计算原始数据的特征向量即可实现数据的特征提取，同时基于原始数据的低维特征向量对原始数据进行表示，实现数据降维。

首先计算企业各风险指标的协方差矩阵如表5.6所示：

表5.6 企业风险指标协方差矩阵

	销售规模（对数）	销项开票数量（对数）	平均销项开票金额（对数）	进购规模（对数）	税价比	发票平均税率	下游客户数量	指定周期负数发票金额比率	固定周期下游客户数量	销方基尼系数	销方HIHI指数
销售规模（对数）	1.000	.762	.683	.083	.308	.299	.186	.088	.186	.521	-.559
销项开票数量（对数）	.762	1.000	.048	.083	.287	.282	.491	.059	.491	.611	-.612
平均销项开票金额（对数）	.683	.048	1.000	.035	.152	.143	-.266	.069	-.266	.114	-.172
进购规模（对数）	.083	.083	.035	1.000	.175	.173	.004	-.119	.004	.045	-.076
税价比	.308	.287	.152	.175	1.000	.999	-.010	-.052	-.010	.215	-.220
发票平均税率	.299	.282	.143	.173	.999	1.000	-.006	-.055	-.006	.213	-.217
下游客户数量	.186	.491	-.266	.004	-.010	-.006	1.000	-.019	1.000	.119	-.113
指定周期负数发票金额比率	.088	.059	.069	-.119	-.052	-.055	-.019	1.000	-.019	.103	-.093
固定周期下游客户数量	.186	.491	-.266	.004	-.010	-.006	1.000	-.019	1.000	.119	-.113
销方基尼系数	.521	.611	.114	.045	.215	.213	.119	.103	.119	1.000	-.914
销方HIHI指数	-.559	-.612	-.172	-.076	-.220	-.217	-.113	-.093	-.113	-.914	1.000

如上表所示可以看到销售规模、销项开票数、平均销项开票金额，销方基尼系数、销方HIHI指数等指标相关性较大。同时主成分其特征值和累计贡献率及成分矩阵如表5.7所示：

表5.7 企业风险指标的特征值和累计贡献率

主成份	初始特征值			提取载荷平方和		
	总计	% 方差百分比	累积 %	总计	% 方差百分比	累积 %
1	485373.630	99.997	78.997	485373.630	99.997	99.997
2	6.249	11.001	99.999			
3	5.241	.001	100.000			
4	1.270	.000	100.000			
5	.004	7.924E-07	100.000			
6	.003	6.223E-07	100.000			
7	.001	2.103E-07	100.000			
8	.000	3.352E-08	100.000			
9	9.953E-07	2.051E-10	100.000			
10	8.444E-12	1.740E-15	100.000			
11	-1.620E-15	-3.337E-19	100.000			

由表5.7可以看出，主成分1+2的累积贡献率以接近100%，其能很好表示原始数据的特征。最后基于PCA的成分矩阵如下：

表5.8 主成分与各指标关系

主成	销售规模	销项开票数量	平均销项开票	进购规模	税价	发票平均	下游客户	指定周期负数	固定周期下游	销方基尼	销方HIHI

分	（ 对数）	（对数）	金额(对数）	（ 对数）	比	税率	数量	发 票 金额比率	客 户 数量	系数	指数
1	.375	.375	-.348	.009	.000	.000	492.632	-.001	492.632	.008	-.004
2	.542	.361	.812	.0.000	.001	.000	-.236	2.365	2.102	.013	-.020

由表 5.8 所示，对于主成分贡献较大的指标包括销售规模、销售开票数量、平均销项开票金额、下游客户。

5.2.2 基于深度神经网络的企业信誉自动识别

2006 年 Hinton 在 Science 上发文[6]，指出利用 RBM 编码预训练深度神经网络与 PCA 相比在高维特征抽取方面有更佳的性能，即深度网拥有强大的特征提取能力。同时还指出深层的网络训练可以通过逐层训练的方式实现，这也使得更深网络的训练成为了可能。虽然这篇文章现在看来并没有在理论上做出较大的创新，尤其是逐层训练的方式早已被弃用，但是该文章却使得深度学习重新回到人们的视野。与此同时随着计算机技术的进步以及互联网的普及，其为深度学习技术提拱了爆发向前的燃剂。真正使深度学习得到广泛关注的是在 2012 年 ImageNet[7] 比赛中，当时 Hinton 组凭借 AlexNet 深度神经网络以领先第二名 10.8 个百分点的优势一举多得比赛冠军。在 Alexnet 中，ReLu 激活函数的使用克服了梯度消失的问题。Dropout 操作将网络的某些连接随机置零有效减小了过拟合的发生，同时利用 GPU，通过并行计算大大缩短了网络的训练时间。接下来一年的 ImageNet 竞赛中，前十的方案几乎全部选择了深度网络的方案。此后新的深度神经网络不断被提出、改进，如 GoogLeNet[8]、VGGNet[9]、ResNet[10]、GAN[11]等等，网络向更深、更复杂的方向发展。在其它的领域如语音识别、机器翻译、问答系统等某些十分微小具体的方面，深度网络也表现出了准确率优于人类的良好性能。一般，可认为网络层数超过三层的网络即为深度神经网络。

（1）模型结构

本文构建的深度神经网络结构如下图所示：

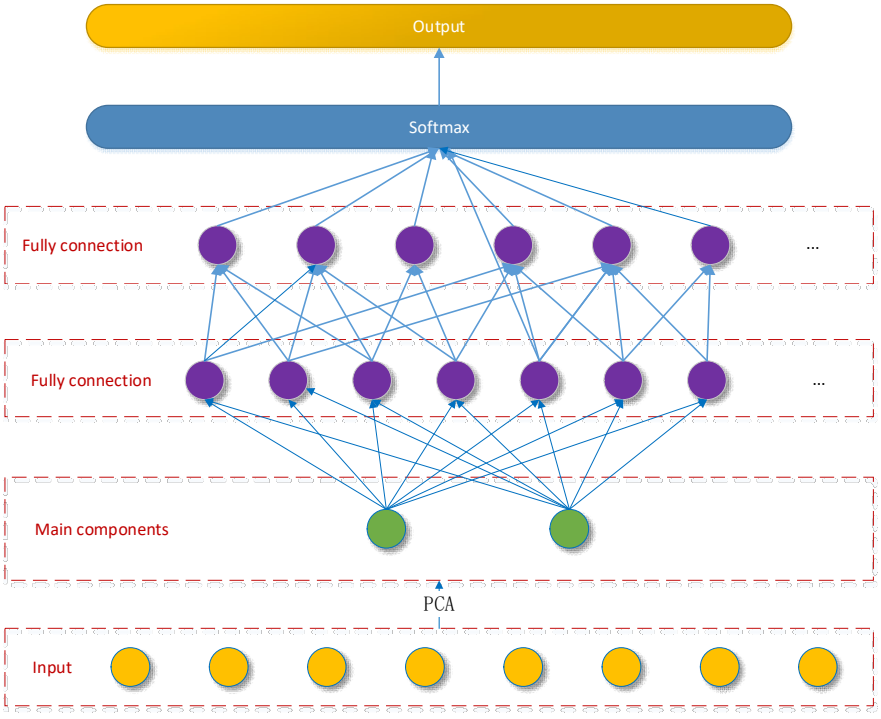

图 5.7 深度神经网络结构示意图

如上图所示，该模型主要包括三个部分：

①首先对原始模型输入数据即企业的风险等级评价指标进行 PCA 处理提取主成分；

②将主成分结果送入至深度神经网络的全连接层中进行特征表示；

③由 Softmax 函数计算各企业信誉等的概率分布；

为防止梯度消失，模型选择 ReLU 为激活函数，其公式如下：

$$f(x)=\max(0,w^{T}x+b) \tag{3}$$

上式中，w^{T},b 分别表示权值和偏置，其为模型的超参数由训练获得，该值的大小即反映模型对不同特征的关注程度。x 为网络输入。

Softmax 函数如下所示：

$$P(y=j\mid x)=\frac{e^{x^{T}w_{j}}}{\sum_{k=1}^{K}e^{x^{T}w_{k}}} \tag{4}$$

如式（5-12）所示，Softmax 函数最终输出即为样本属于各个类别的概率分布，其中 $x^{T}w$ 可视为经过神经网络全连接层后的输出。

模型的目标函数如下：

$$L=-\sum_{k=1}^{n}\sum_{i=1}^{C}p_{ki}\log\left(\hat{p}_{ki}\right)+\lambda\left\|W\right\|_{2}^{2} \tag{5}$$

上式中，p_{ki} 为样本 k 属于第 i 类的概率；

$\hat{p}_{ki}$ 为网络预测样本 k 属于第 i 类的概率；

λ 为正则化权值。

模型选择交叉熵损失作为目标函数，同时为减小过拟合的风险这里引入二范数进行正则化操作。

（2）数据集的准备

由附件一知，共有 120 余组企业信息数据，本文随机从中抽取 100 组数据作为验证集，其它 20 多组数据作为训练集训练网络。首先对原始数据进行标准化处理，如下：

$$x_{std}=\frac{x-\bar{x}}{s},\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_{i},s=\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_{i}-\bar{x})^{2}} \tag{6}$$

上式中，$\bar{x}$ 表示均值，s 表示标准差。通过数据标准化处理消除数据量纲的影响，同时使得模型训练更加容易。

（3）模型的评价

选择准确率、查全率、F1 值和 ROC（receiver operating characteristic curve）曲线的 AUC 值作为模型的评价指标，如下：

$$P=\frac{TP}{TP+FP},\quad R=\frac{TP}{TP+FN},\quad F_{1}=\frac{2P\bullet R}{P+R} \tag{7}$$

上式中，TP、FP、FN 分别代表真正例、假正例和假反例。

（4）模型的设置

利用 10 次 10 折交叉训练对网络进行训练，利用测试样本评价网络模型的性能，深度神经网络权连接层神经元数目均为 128，层数为 3。使用随机梯度下降进行训练，学习率为，迭代次数为 25。

（5）结果分析

深度神经网络的模型训练和测试结果如图 5.8 所示：

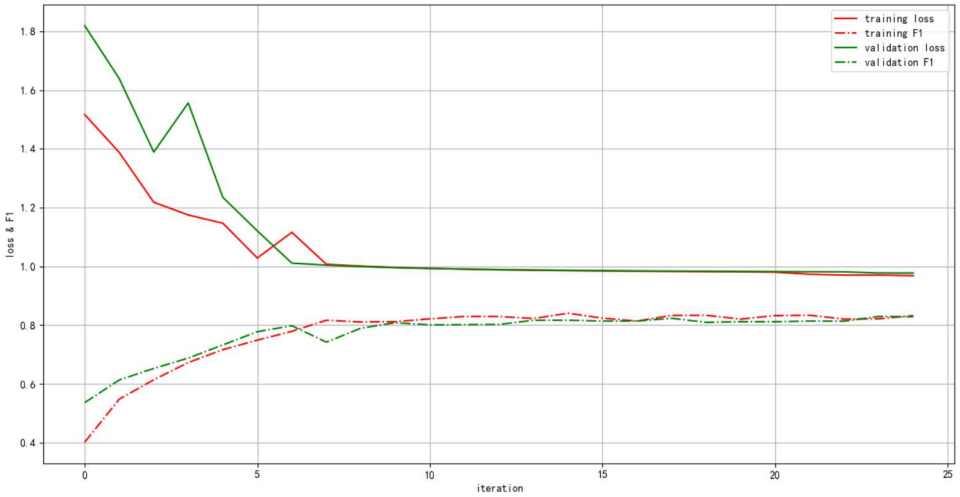

图 5.8 深度神经网络训练过程 loss 及 F1 值

由图 5.8 可以明显看出模型在迭代至 10 轮左右就已经收敛，且 F1 值保持在 0.8 左右。

基于训练的神经网络对附件二所给企业进行信誉等级识别，附件三个信誉等级的企业数目如下：

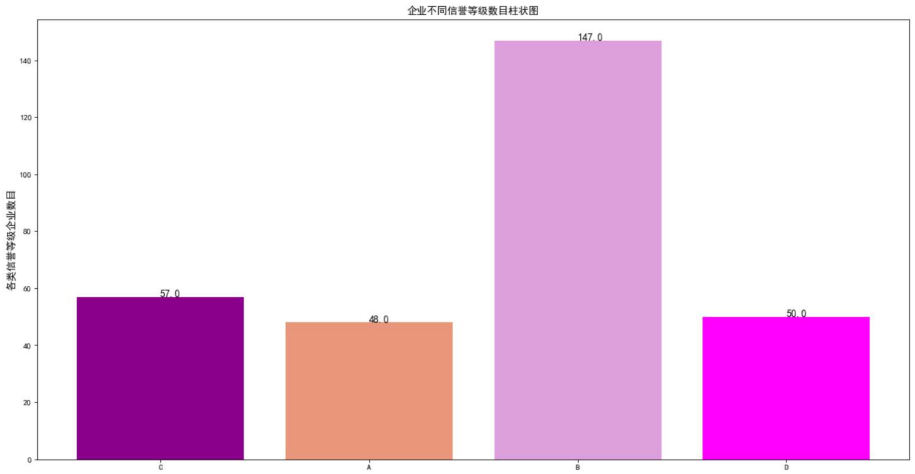

图 5.9 附件二企业信誉等级识别结果分布柱状图

如图 5.9 所示，可以看到附件二所给的 302 家企业中有近一半企业的信誉等级为 B，而其它三类信誉等级的企业数目大致相同。

5.2.3 考虑顾客流失的银行信贷策略建模

由附件三知不同信誉等级的企业其在不同借贷年利率下流失率即可能性大小也将不同，因此本文先讨论不同信誉等级下客户流失率与借贷利率间的关系，然后根据客户流失率对式（2）进行修正，获得银行总借贷金额确定时不同企业的借贷策略。

（1）不同信誉等级借贷利率与客户流失率关系建模

本文考虑使用指数函数对不同信誉等级借贷利率与客户流失率进行拟合，设计指数函数如下：

$$y_{loss} = a x_{rate}^{\ b} + c \qquad (8)$$

上式中，x_{rate}, y_{loss} 分别表示借贷利率与客户流失率；

a,b,c 为拟合超参数；

本文使用最小二乘方法拟合，即最小化均方误差，如下：

$$\min f(y,y_{loss})=\frac{1}{n}\sum_{i=1}^{n}\left(y_i-y_{loss_i}\right)^2 \tag{9}$$

上式中，y_i,y_{loss_i} 分别表示实际坐标和函数拟合坐标。拟合所得曲线如下图所示：

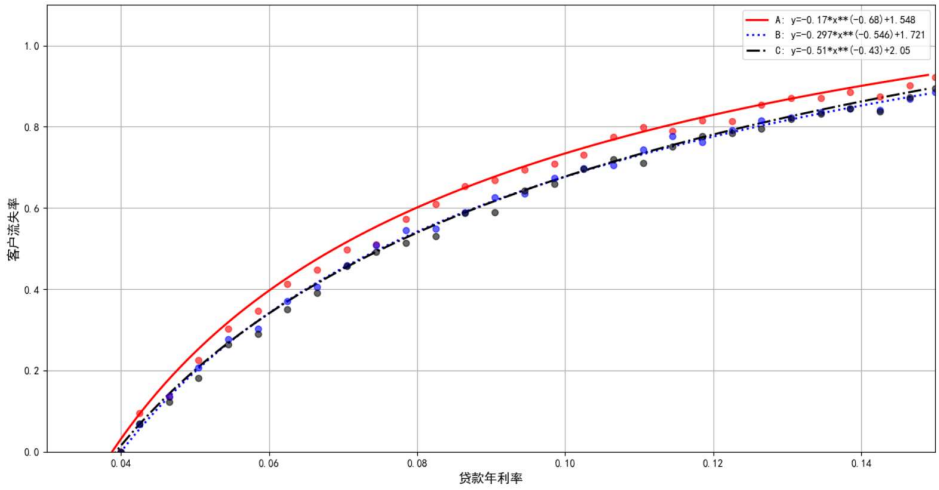

图 5.10 企业不同信誉等级下贷款年利率与客户流失率拟合曲线

拟合函数如下表所示：

表 5.9 企业不同信誉等级下贷款年利率与客户流失率拟合函数

企业信誉等级	拟合函数	RMSE
A	$y=-0.1716x^{-0.6825}+1.548$	0.01317
B	$y=-0.2966x^{-0.546}+1.721$	0.01044
C	$y=-0.5104x^{-0.4313}+2.045$	0.01211

结合图 5.10 和表 5.9 所示，可以看出对于 A，B，C 三类信誉等级企业其贷款年利率与客户流失率规律基本相同，且使用指数函数进行拟合其误差较小，故该函数能很好反映企业不同信誉等级下贷款年利率与客户流失率的规律。

（2）考虑企业流失的借贷策略修正

如图 5.10 可以明显看出，当借贷年利率增加时客户的流失率也呈上升趋势，因此这里考虑对 5.1 小结中的公式（2）进行调整，增加客户流失的潜在损失，如下：

$$P=\sum_{i=1}^{n}\left((1+\lambda_i)x_i-\gamma f_{loss}(\lambda_i)x_i\right) \tag{10}$$

上式中，x_i 表示企业 i 的获贷金额；

λ_i 表示企业 i 的获贷年利率；

$f_{loss}(\lambda_i)$ 表示年利率为 λ_i 时的企业流失率；

γ 为银行获利损失调节因子；

故 $\gamma f_{loss}(\lambda_i)x_i$ 即表示银行因贷款利率所造成的企业流失带来的损失，式（10）的约束仍与式（2）保持一致，求解最优（最大）P 获得各个企业的贷款策略如下：

表 5.10 不同企业借贷策略（截取部分）

企业代号	信誉评级	借贷金额（万元）	借贷年利率（%）
E124	C	27.5	4
E125	C	15.91	4
E126	A	57.12	12
E127	C	25.28	6
E128	A	54.6	10
E129	C	13.16	4
E130	C	12.7	7
E131	B	43.25	7
E132	B	58.68	6
E133	A	64.12	14
E134	B	34.81	7

不同企业的借贷策略如表 5.10 所示，完成结果见附件。可以看出对于信誉等级为 D 的企业其借贷金额为 0。而对于信誉等级分别为 B，C，D 的企业其借贷金额分别集中在 30,50 和 70 的附近，具体金额还需由企业自身经营管理状况即税票信息确定。

5.3 突发情形下的企业信贷策略建模研究

在实际情况中某些突发因素，如严重事故、灾难、重大战略或政策调整将对企业的经营管理考虑造成直接的影响[13][14]。因此，企业的违约意愿和还款能力也将受到巨大冲击。如何在突发因素的作用下，银行迅速调整借贷策略以尽可能的保证收益、减小损失，该研究具有重要的意义。

5.3.1 突发因素下考虑企业性质及规模的借贷策略建模

本文首先假设突发因素将对所有企业的经营状况带来一定的影响，而该影响的性质（“积极”或“消极”）以及作用的大小，主要由企业的类型和历史经营状况决定。如对于新冠病毒，若该企业为医药及医药建材型企业则影响积极，若为娱乐型产业则影响消极。同时，若该企业历史经营状况良好，则影响程度将较小，若该企业历史经营状况较差，则影响程度较大。因此，本文首先考虑对影响积极且经营状况良好的企业可适当增加贷款额度，而对于影响消极、经营状况较差的企业需减少贷款额度。同时对利率进行调整，以使得银行获利最大、风险最低。

综上所述，本文设计企业贷款金额修正模型，主要考虑如下两点：

（1）突发因素的严重程度。不同突发因素所造成的后果不同，若该突发事件造成持久的全球性影响则影响程度大，若为区域性短时影响，则影响程度将较小；

（2）突发因素对企业的性质。不同突发因素对不同类型企业的影响性质也将不同，某一突发因素可能对于部分企业产生积极影响，而对其它企业产生负面作用，其与企业及突发因素自身特性有关；

（3）突发因素对企业的影响程度。不同规模和经营状况的企业在相同突发因素的冲击下受到的影响程度也将不同，其与本身企业抗打击能力有关；

故本文考虑以上三点，设计贷款金额修正函数如下：

$$\begin{aligned} x^{'} &= x + vdm(x) \\ m(x) &= -\frac{x}{1+e^{-\varsigma \log(m_{out}/m_{in})}} + 1 \end{aligned} \tag{11}$$

上式中，x'为修正后的贷款金额，x为原始贷款金额；

v表示突发因素对企业的性质，其取值为± 1；

d表示突发因素的严重程度，该值的确定如表 5.11 所示；

ς为修正因子；

$m(x)$表示突发因素对企业的影响程度，其取值有企业经营状况，即销项与进项发票面额之差决定；

表 5.11 突发因素严重程度表

影响范围	影响时间	影响程度d
全球	长	5
	中	4.5
	短	4
地区	长	3.5
	中	3
	短	2
局部	长	2
	中	1.5
	短	1

如表 5.11 所示，突发事件的严重程度主要与影响范围和作用时间这两个因素有关，若作用范围为全球且时间长，则影响程度最大d取 5。若作用范围为局部且影响时间短，则影响程度最小d取 1。

超参数ς分别取 2,1,0.5 时，企业受影响程度与企业规模即$\log(m_{out}/m_{in})$的变化曲线如下：

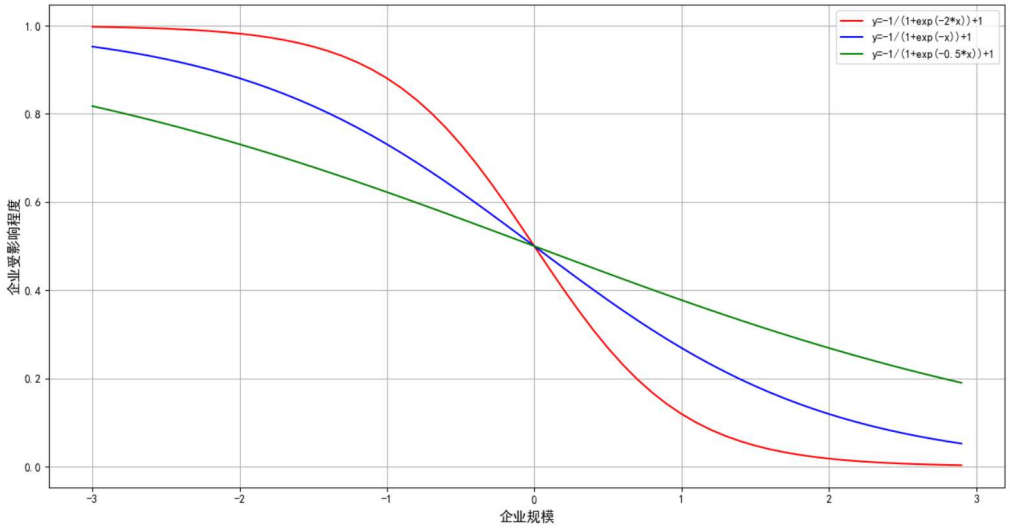

图 5.11 不同ς下突发因素对企业的影响程度曲线

可以看出突发因素对企业的影响程度曲线的取值范围为 0～1，且随着企业规模的增加其影响程度逐渐下降。

5.3.2 以新冠肺炎为例企业信贷策略的调整策略研究

考虑 Conv-19 的对全球产业链和经济造成的影响，这里将其严重程度设为最高即$d=5$。依据附件二所给企业名称分析其企业类型，判断新冠疫情对该行业是否为正面影响。分析知在所给 302 家企业中共有 8 家企业（E195,E197,E251,E261,E321,E379,E398,E420）与健康、医疗和生物有关，这里假设新冠疫情对该企业呈正面影响，其他 294 家企业均为负面影响。

结合式（11）和式（10）求解在不同修正因子ς下的企业贷款策略，如下：

表 5.12 不同修正因子ς下的企业贷款策略（截取部分）

企业代号	信誉评级	借贷金额（万元，0.5）	借贷金额（万元，1）	借贷金额（万元，2）	借贷年利率（%）
E124	C	10	10	10	4
E125	C	10	12.9	17.66	4
E126	A	57.53454997	60.7	49.05	11.83482269
E127	C	10	10	10	4
E128	A	100	96.28	98.12	9.676814635
E129	C	10	10	13.27	4
E130	C	10	12.56	12.48	4.895261104
E131	B	98.73233241	99.43	91.12	6.838776318
E132	B	76.52744297	76.79	78.29	5.9608326
E133	A	100	98.05	96.53	13.57748196
E134	B	100	100	100	6.57372012
E135	B	31.02180665	32.16	21.96	7.299134707
E136	B	68.87439928	67.82	64.78	9.121826541
E137	B	100	100	100	6.711500195
E138	C	25.76819335	24.81	18.93	4
E139	B	76.83093628	72.7	74.22	8.261347119
E140	B	52.66704514	49.88	54.29	6.262528689

如上表所示，可以看到对于不同修正因子下其对企业贷款策略的作用不同，修正因子ς越大，整体而言其贷款金额调整的幅度也降越大，策略偏“激进”。而ς越小，其贷款金额的调整幅度也将越小，策略偏“保守”。同时，对于信誉等级为 D 的企业其贷款均不准入，此外借贷年利率在新冠病毒的作用下整体向下调整，以缓解企业的资金难问题。

六、模型的评价与推广

6.1 模型的评价

6.1.1 结合熵值取权层次分析法模型评价

层次分析法在工程实际中应用广发，操作简单且可解释性好。但是权值的确定带有极大的主观经验。因此本文考虑使用熵值法对 AHP 所得权值进行修正，以消除主观偏见。在后续的研究中可以考虑如基于随机交叉博弈的数据包络方法实现客观赋权。

6.1.2 考虑企业信誉、规模和客户流失的信贷策略模型评价

在对不同企业进行信贷方案建模时，主要考虑企业的信誉评级、企业的进行规模和年代利率对用户流失的影响。通过该模型可以实现针对不同规模企业的信贷方案个性化设计，而非简单的考虑企业分级确定信贷策略。该模型适用范围更广且策略设计粒度更精细。但是，本文对于企业经营状态的衡量仅考虑企业进销发票总额数据，该方法还需进一步补充完善。

6.1.3 基于深度神经网络的企业信誉等级自动识别模型评价

利用深度神经网络强大的特征表示能力可以实现企业信誉等级的自动识别，为后续企业信贷策略的指定提供有力支撑。目前神经网络模型识别精度较低，仍需进一步训练和调优。

6.1.4 突发因素下企业借贷方案调整策略模型评价

本文分别针对不同企业类型、不同突发因素严重程度、突发因素对企业的影

响类型（积极或消极）以及企业的经营状况对借贷放案进行调整。该模型考虑全面，鲁棒性强。能适用于不同规模、类型企业、不同突发因素下的借贷方案动态调整。但是，对于突发因素严重程度的判断仍较为主观，后续研究中课引入模糊数学隶属函数的思想进行修正。

6.2 模型的推广

6.2.1 融合熵值取权的层次分析法模型

该模型对于复杂系统的评价问题均适用，尤其适用主观性较强，缺少理化指标的系统。如企业经营状况的评价、人员素质的评价、教育质量的评价、影响力评价等。

6.2.1 基于深度神经网络的企业信誉等级识别模型

该模型本质上为多分类模型，对于数值型输入数据的分类问题该模型均可以进行迁移泛化，如水质的分类、基于理化指标的葡萄酒品质分类等。同时对该模型进行一定的改进如将最后的 Softmax 函数替换为神经元等即可解决回归问题，如人口的预测等。

七、参考文献

[1] 万钰.H 银行 B 分行中小企业信贷风险管理研究[D].北京交通大学,2019.
[2] 郭金玉,张忠彬,孙庆云.层次分析法的研究与应用[J].中国安全科学学报,2008,18(5):148-153.
[3] 邓雪,李家铭,曾浩健,等.层次分析法权重计算方法分析及其应用研究[D].2012.
[4] 王兰化,张莺.层次分析-熵值定权法在城市建设用地适宜性评价中的应用[J].地质调查与研究,2011,34(4): 305-312.
[5] Abdi. H., & Williams, L.J. Principal component analysis.. Wiley Interdisciplinary Reviews: Computational Statistics,. 2010, 2: 433－459.
[6] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural computation, 2006, 18(7): 1527-1554.
[7] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database[C]//2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009: 248-255.
[8] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
[9] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
[10] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
[11] Creswell A, White T, Dumoulin V, et al. Generative adversarial networks: An overview[J]. IEEE Signal Processing Magazine, 2018, 35(1): 53-65.
[12] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model[J]. Journal of machine learning research, 2003, 3(Feb): 1137-1155.
[13]张宇. 工商银行威海分行突发性风险管理研究[D].哈尔滨工业大学,2018.
[14] 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.

八、附录

8.1 指标体系设计

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

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

import plotly.graph_objects as go


def pie_chart(path_raw):
    raw_data_comp = pd.read_excel(path_raw, encoding='utf_8_sig',
sheet_name='企业信息')
    temp_dict = dict(Counter(raw_data_comp['是否违约']))
    labels = list(temp_dict.keys())
    values = list(temp_dict.values())
    fig = go.Figure(data=[go.Pie(labels=labels, values=values,
textinfo='label+percent',
                                 insidetextorientation='radial')])
    fig.show()


def hot_map(cosin_epoch):
    y_grid = ['是', '否']
    x_grid = ['A', 'B', 'C', 'D']
    # 定义画布为 1*1 个划分，并在第 1 个位置上进行作图
    # array_save = np.array(cosin_epoch).transpose()
    sns.heatmap(cosin_epoch, annot=cosin_epoch, fmt=".2g",
cmap="RdBu_r", xticklabels=x_grid, yticklabels=y_grid)
    plt.xlabel("信誉评级", fontsize=12.5)
    plt.ylabel("是否违约", fontsize=12.5)
    plt.title("信誉评级与是否违约的联合概率", fontsize=12.5)
    plt.show()
```

```
def correlation(path_raw):
    raw_data_comp  =  pd.read_excel(path_raw,  encoding='utf_8_sig',
sheet_name='企业信息')
    a_1 = 0
    b_1 = 0
    c_1 = 0
    d_1 = 0
    a_0 = 0
    b_0 = 0
    c_0 = 0
    d_0 = 0
    for i in range(len(raw_data_comp)):
        if  raw_data_comp.iloc[i]['是否违约']  ==  '是'  and
raw_data_comp.iloc[i]['信誉评级'] == 'A':
            a_1 += 1
        elif  raw_data_comp.iloc[i]['是否违约']  ==  '是'  and
raw_data_comp.iloc[i]['信誉评级'] == 'B':
            b_1 += 1
        elif  raw_data_comp.iloc[i]['是否违约']  ==  '是'  and
raw_data_comp.iloc[i]['信誉评级'] == 'C':
            c_1 += 1
        elif  raw_data_comp.iloc[i]['是否违约']  ==  '是'  and
raw_data_comp.iloc[i]['信誉评级'] == 'D':
            d_1 += 1
        elif  raw_data_comp.iloc[i]['是否违约']  ==  '否'  and
raw_data_comp.iloc[i]['信誉评级'] == 'A':
            a_0 += 1
        elif  raw_data_comp.iloc[i]['是否违约']  ==  '否'  and
raw_data_comp.iloc[i]['信誉评级'] == 'B':
            b_0 += 1
        elif  raw_data_comp.iloc[i]['是否违约']  ==  '否'  and
raw_data_comp.iloc[i]['信誉评级'] == 'C':
            c_0 += 1
        elif  raw_data_comp.iloc[i]['是否违约']  ==  '否'  and
raw_data_comp.iloc[i]['信誉评级'] == 'D':
            d_0 += 1
    break_dict = dict(Counter(raw_data_comp['是否违约']))
    repu_dict = dict(Counter(raw_data_comp['信誉评级']))
    a_1_p = round(a_1 / (break_dict['是'] + repu_dict['A'] - a_1), 4)
    b_1_p = round(b_1 / (break_dict['是'] + repu_dict['B'] - b_1), 4)
    c_1_p = round(c_1 / (break_dict['是'] + repu_dict['C'] - c_1), 4)
    d_1_p = round(d_1 / (break_dict['是'] + repu_dict['D'] - d_1), 4)
    a_0_p = round(a_0 / (break_dict['否'] + repu_dict['A'] - a_0), 4)
```

```
    b_0_p = round(b_0 / (break_dict['否'] + repu_dict['B'] - b_0), 4)
    c_0_p = round(c_0 / (break_dict['否'] + repu_dict['C'] - c_0), 4)
    d_0_p = round(d_0 / (break_dict['否'] + repu_dict['D'] - d_0), 4)
    hot_map(np.array([[a_1_p, b_1_p, c_1_p, d_1_p], [a_0_p, b_0_p,
c_0_p, d_0_p]]))


def bbox(list_value, list_name):
    y = np.transpose(np.array(list_value))
    labels = list_name
    plt.boxplot(y, labels=labels, sym='o')
    plt.xlabel('指标', fontsize=12.5)
    plt.ylabel('指标值', fontsize=12.5)
    plt.title("企业风险评估指标箱型图", fontsize=12.5)
    plt.grid(True)
    plt.show()


def gini(money_list):
    money_list_sort = sorted(money_list, reverse=True)[1:]
    all_money = sum(money_list_sort)
    return   round(1-(1/(len(money_list_sort)+1))*(2*sum([i/all_money
for i in money_list_sort])+1), 5)


def hini(money_list):
    money_list_sort = sorted(money_list, reverse=True)
    total_all = sum(money_list_sort)
    hini_index = 0
    for i in money_list_sort[:min(50, len(money_list_sort))]:
        hini_index += (i/total_all)**2
    return round(hini_index, 5)


def    sell_all(sell_money_dict,    sell_tax_dict,    sell_all_dict,
sell_comp_dict, sell_status_dict):
    sell_scale = {}
    sell_num = {}
    sell_average = {}
    sell_tax_all_ratio = {}
    sell_tax_money_ratio = {}
    num_cust = {}
    sell_negative_dict = {}
    wrong_dict = {}
```

```
    gini_dict = {}
    hini_dict = {}
    for comp_temp in sell_money_dict:
        sell_scale[comp_temp]                                          =
round(np.log(sum(sell_money_dict[comp_temp])), 5)
        sell_num[comp_temp]                                            =
round(np.log(len(sell_money_dict[comp_temp])), 5)
        sell_average[comp_temp]                                        =
round(np.log(sum(sell_money_dict[comp_temp])                           /
len(sell_money_dict[comp_temp])), 5)
        sell_tax_all_ratio[comp_temp]                                  =
round(sum(sell_tax_dict[comp_temp])  /  sum(sell_all_dict[comp_temp]),
5)
        sell_tax_money_ratio[comp_temp]                                =
round(sum(sell_tax_dict[comp_temp]) / sum(sell_money_dict[comp_temp]),
5)
        num_cust[comp_temp]                                            =
len(list(set(sell_comp_dict[comp_temp])))
        sell_negative_dict[comp_temp] = round(
            sum([abs(i) for i in sell_all_dict[comp_temp] if i < 0])
/ sum(sell_all_dict[comp_temp]), 5)
        wrong_dict[comp_temp] = round(sum(
            [abs(sell_all_dict[comp_temp][index_temp]) for index_temp,
status_temp in enumerate(sell_status_dict) if
             status_temp    ==    '    作    废    发    票    ']))    /
sum(sell_all_dict[comp_temp]), 5)
        gini_dict[comp_temp] = gini(sell_all_dict[comp_temp])
        hini_dict[comp_temp] = hini(sell_all_dict[comp_temp])
    return  sell_scale,  sell_num,  sell_average,  sell_tax_all_ratio,
sell_tax_money_ratio,   num_cust,   sell_negative_dict,   wrong_dict,
gini_dict, hini_dict


def buy_all(sell_money_dict):
    sell_scale = {}
    sell_num = {}
    for comp_temp in sell_money_dict:
        sell_scale[comp_temp]                                          =
round(np.log(sum(sell_money_dict[comp_temp])), 5)
        sell_num[comp_temp]                                            =
round(np.log(len(sell_money_dict[comp_temp])), 5)
    return sell_scale, sell_num
```

```
def index_system_in(path_raw, path_save):
    raw_data_sell = pd.read_csv(path_raw, encoding='utf_8_sig',
engine='python')
    sell_comp_dict = {}
    sell_money_dict = {}
    sell_tax_dict = {}
    sell_all_dict = {}
    sell_status_dict = {}
    for i in range(len(raw_data_sell)):
        if raw_data_sell.iloc[i]['企业代号'] in sell_comp_dict:
            sell_money_dict[raw_data_sell.iloc[i]['企业代号
']].append(raw_data_sell.iloc[i]['金额'])
            # sell_comp_dict[raw_data_sell.iloc[i]['企业代号
']].append(raw_data_sell.iloc[i]['销方单位代号'])
            # sell_tax_dict[raw_data_sell.iloc[i]['企业代号
']].append(raw_data_sell.iloc[i]['税额'])
            # sell_all_dict[raw_data_sell.iloc[i]['企业代号
']].append(raw_data_sell.iloc[i]['价税合计'])
            # sell_status_dict[raw_data_sell.iloc[i]['企业代号
']].append(raw_data_sell.iloc[i]['发票状态'])
        else:
            sell_money_dict[raw_data_sell.iloc[i]['企业代号']] =
[raw_data_sell.iloc[i]['金额']]
            # sell_comp_dict[raw_data_sell.iloc[i]['企业代号']] =
[raw_data_sell.iloc[i]['销方单位代号']]
            # sell_tax_dict[raw_data_sell.iloc[i]['企业代号']] =
[raw_data_sell.iloc[i]['税额']]
            # sell_all_dict[raw_data_sell.iloc[i]['企业代号']] =
[raw_data_sell.iloc[i]['价税合计']]
            # sell_status_dict[raw_data_sell.iloc[i]['企业代号']] =
[raw_data_sell.iloc[i]['发票状态']]
    buy_scale, buy_num = buy_all(sell_money_dict)
    pd.DataFrame({'进购规模（对数）': buy_scale, '进项开票数量（对数）
': buy_num}).to_csv(path_save, encoding='utf_8_sig', index=False)


def index_system_out(path_raw, path_save):
    # pd.read_excel(path_raw, encoding='utf_8_sig', sheet_name='进项
发票信息').to_csv(path_save, encoding='utf_8_sig', index=False)


    raw_data_sell = pd.read_csv(path_raw, encoding='utf_8_sig',
engine='python')
    # raw_data_pru = pd.read_excel(path_raw, encoding='utf_8_sig',
```

```
sheet_name='进项发票信息')
    sell_comp_dict = {}
    sell_money_dict = {}
    sell_tax_dict = {}
    sell_all_dict = {}
    sell_status_dict = {}
    for i in range(len(raw_data_sell)):
        if raw_data_sell.iloc[i]['企业代号'] in sell_comp_dict:
            sell_comp_dict[raw_data_sell.iloc[i]['企业代号']].append(raw_data_sell.iloc[i]['购方单位代号'])
            sell_money_dict[raw_data_sell.iloc[i]['企业代号']].append(raw_data_sell.iloc[i]['金额'])
            sell_tax_dict[raw_data_sell.iloc[i]['企业代号']].append(raw_data_sell.iloc[i]['税额'])
            sell_all_dict[raw_data_sell.iloc[i]['企业代号']].append(raw_data_sell.iloc[i]['价税合计'])
            sell_status_dict[raw_data_sell.iloc[i]['企业代号']].append(raw_data_sell.iloc[i]['发票状态'])
        else:
            sell_comp_dict[raw_data_sell.iloc[i]['企业代号']] = [raw_data_sell.iloc[i]['购方单位代号']]
            sell_money_dict[raw_data_sell.iloc[i]['企业代号']] = [raw_data_sell.iloc[i]['金额']]
            sell_tax_dict[raw_data_sell.iloc[i]['企业代号']] = [raw_data_sell.iloc[i]['税额']]
            sell_all_dict[raw_data_sell.iloc[i]['企业代号']] = [raw_data_sell.iloc[i]['价税合计']]
            sell_status_dict[raw_data_sell.iloc[i]['企业代号']] = [raw_data_sell.iloc[i]['发票状态']]
    sell_scale, sell_num, sell_average, sell_tax_all_ratio, sell_tax_money_ratio, num_cust, sell_negative_dict, \
    wrong_dict, gini_dict, hini_dict = sell_all(sell_money_dict, sell_tax_dict, sell_all_dict, sell_comp_dict, sell_status_dict)
    pd.DataFrame({'销售规模（对数）':sell_scale, '销项开票数量（对数）':sell_num, '平均销项开票金额（对数）':sell_average, '税价比':sell_tax_all_ratio,
                  '发票平均税率':sell_tax_money_ratio, '下游客户数量':num_cust, '指定周期负数发票金额比率':sell_negative_dict,
                  '指定周期作废发票金额比率':wrong_dict, '固定周期下游客户数量':num_cust, '销方基尼系数':gini_dict, '销方 HIHI 指数':hini_dict}).to_csv(path_save, encoding='utf_8_sig', index=False)
```

```
    #    bbox([list(sell_scale.values()),    list(sell_num.values()),
list(sell_average.values())],
    #      ['销售规模（对数）', '销售开票数量（对数）', '平均销售开票
金额（对数）'])


def score_comp(path_index, path_save):
    raw_data    =    pd.read_csv(path_index,    encoding='utf_8_sig',
engine='python')
    positive_index = ['销售规模（对数）', '销项开票数量（对数）', '平
均销项开票金额（对数）', '进购规模（对数）',
                      '进项开票数量（对数）', '税价比', '发票平均税率
', '下游客户数量', '固定周期下游客户数量']
    negative_index = ['指定周期负数发票金额比率', '指定周期作废发票金
额比率', '销方基尼系数', '销方 HIHI 指数']
    class_index = ['信誉评级']
    bi_index = ['是否违约']
    dict_score = {}
    dict_score['企业代号'] = list(raw_data['企业代号'])
    for positive_temp in positive_index:
        dict_score[positive_temp] = []
        list_temp = list(raw_data[positive_temp])
        min_temp = min(list_temp)
        max_temp = max(list_temp)
        temp_20 = np.percentile(list_temp, 20)
        temp_40 = np.percentile(list_temp, 40)
        temp_60 = np.percentile(list_temp, 60)
        temp_80 = np.percentile(list_temp, 80)
        for val_temp in list_temp:
            if min_temp <= val_temp < temp_20:
                dict_score[positive_temp].append(1)
            elif temp_20 <= val_temp < temp_40:
                dict_score[positive_temp].append(2)
            elif temp_40 <= val_temp < temp_60:
                dict_score[positive_temp].append(3)
            elif temp_60 <= val_temp < temp_80:
                dict_score[positive_temp].append(4)
            elif temp_80 <= val_temp <= max_temp:
                dict_score[positive_temp].append(5)

    for negative_temp in negative_index:
        dict_score[negative_temp] = []
        list_temp = list(raw_data[negative_temp])
```

```
        min_temp = min(list_temp)
        max_temp = max(list_temp)
        temp_20 = np.percentile(list_temp, 20)
        temp_40 = np.percentile(list_temp, 40)
        temp_60 = np.percentile(list_temp, 60)
        temp_80 = np.percentile(list_temp, 80)
        for val_temp in list_temp:
            if min_temp <= val_temp < temp_20:
                dict_score[negative_temp].append(5)
            elif temp_20 <= val_temp < temp_40:
                dict_score[negative_temp].append(4)
            elif temp_40 <= val_temp < temp_60:
                dict_score[negative_temp].append(3)
            elif temp_60 <= val_temp < temp_80:
                dict_score[negative_temp].append(2)
            elif temp_80 <= val_temp <= max_temp:
                dict_score[negative_temp].append(1)
    dict_score['信誉评级'] = []
    for level_temp in list(raw_data['信誉评级']):
        if level_temp == 'A':
            dict_score['信誉评级'].append(5)
        elif level_temp == 'B':
            dict_score['信誉评级'].append(4)
        elif level_temp == 'C':
            dict_score['信誉评级'].append(2)
        elif level_temp == 'D':
            dict_score['信誉评级'].append(1)
    dict_score['是否违约'] = []
    for level_temp in list(raw_data['是否违约']):
        if level_temp == '否':
            dict_score['是否违约'].append(5)
        elif level_temp == '是':
            dict_score['是否违约'].append(1)

    pd.DataFrame(dict_score).to_csv(path_save,  encoding='utf_8_sig',
index=False)


def comprehensive_score(path_score, path_save):
    weight=
[0.078,0.041,0.065,0.048,0.017,0.041,0.051,0.045,0.037,0.021,0.05,0.0
45,0.025,0.095,0.335]
    raw_data = pd.read_csv(path_score, encoding='utf_8_sig')
    score_all = []
```

```
    for i in range(len(raw_data)):
        score_temp = 0
        for j, weight_temp in zip(list(raw_data.iloc[i])[1:], weight):
            score_temp += float(j) * weight_temp
        score_all.append(round(score_temp, 5))
    pd.DataFrame({'综合得分':score_all}).to_csv(path_save,
encoding='utf_8_sig', index=False)


def bar_plt(dict_main, list_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.hist(list_main,   bins=50,   normed=0,   facecolor="yellow",
edgecolor="black", alpha=0.7)
    # plt.bar(dict_main.keys(), list_count, color=c)
    # for i, val in enumerate(list_count):
    #     plt.text(i, val, float(val), fontdict={'size': 12.5})
    # Decoration
    plt.title("信誉为D企业得分统计柱状图", fontsize=12.5)
    ax1.set_ylabel('得分数目', fontsize=12.5)
    plt.show()


def plot_score(path_raw):
    raw_data = pd.read_csv(path_raw, encoding='utf_8_sig')
    score = list(raw_data['综合得分'])
    level = list(raw_data['信誉评级'])
    score_round = []
    for score_temp in score:
        score_round.append(round(score_temp, 1))
    score_a = []
    score_b = []
    score_c = []
    score_d = []
    for level_temp, score_temp in zip(level, score_round):
        if level_temp == 5:
            score_a.append(score_temp)
        elif level_temp == 4:
            score_b.append(score_temp)
```

```
        elif level_temp == 2:
            score_c.append(score_temp)
        elif level_temp == 1:
            score_d.append(score_temp)
    bar_plt(dict(Counter(score_d)), score_d)


def loss_fig(path_loss):
    raw_data_comp = pd.read_excel(path_loss, encoding='utf_8_sig',
sheet_name='Sheet1')

    x_point_a = list(raw_data_comp['贷款年利率'])[1:]
    y_point_a = list(raw_data_comp['客户流失率'])[1:]
    y_point_b = list(raw_data_comp['Unnamed: 2'])[1:]
    y_point_c = list(raw_data_comp['Unnamed: 3'])[1:]
    x = np.arange(0.03, 0.15, 0.001)
    y_a = -0.17 * x ** (-0.68) + 1.548
    y_b = -0.297 * x ** (-0.546) + 1.721
    y_c = -0.51 * x ** (-0.43) + 2.05
    plt.plot(x, y_a, label="A: y=-0.17*x**(-0.68)+1.548", color="red",
linewidth=2, linestyle='-')
    plt.plot(x, y_b, label="B: y=-0.297*x**(-0.546)+1.721",
color="blue", linewidth=2, linestyle=':')
    plt.plot(x, y_c, label="C: y=-0.51*x**(-0.43)+2.05", color="black",
linewidth=2, linestyle='-.')
    plt.plot(x_point_a, y_point_a, 'ro', alpha=0.6, linewidth=0.1)
    plt.plot(x_point_a, y_point_b, 'bo', alpha=0.6, linewidth=0.1)
    plt.plot(x_point_a, y_point_c, 'ko', 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 strategy_money(path_raw, path_save):
    raw_data = pd.read_csv(path_raw, encoding='utf_8_sig',
engine='python')
    comp_index = list(raw_data['企业代号'])
    level = list(raw_data['信誉评级'])
    strategy_list = []
    strategy_rate_list = []
```

```
    for level_temp in level:
        if level_temp == 'D':
            strategy_list.append(0)
            strategy_rate_list.append(0)
        elif level_temp == 'A':
            strategy_list.append(70+random.randint(-10, 10))
            strategy_rate_list.append(12+random.randint(-2, 2))
        elif level_temp == 'B':
            strategy_list.append(50+random.randint(-10, 10))
            strategy_rate_list.append(8+random.randint(-2, 2))
        if level_temp == 'C':
            strategy_list.append(30+random.randint(-10, 10))
            strategy_rate_list.append(6+random.randint(-2, 2))
    if sum(strategy_list) - 10000>0:
        x0 = np.random.rand(len(raw_data))
        ratio = (sum(strategy_list) - 10000) / sum(x0)
        x1 = x0 * ratio
        for i in range(len(x1)):
            strategy_list[i] = round(strategy_list[i]-x1[i], 2)
    else:
        x0 = np.random.rand(len(raw_data))
        ratio = abs(sum(strategy_list) - 10000) / sum(x0)
        x1 = x0 * ratio
        for i in range(len(x1)):
            strategy_list[i] = round(strategy_list[i] + x1[i], 2)
    pd.DataFrame({'企业代号':comp_index, '信誉评级':level, '借贷金额
（ 万 元 ） ':strategy_list, ' 借 贷 年 利 率
':strategy_rate_list}).to_csv(path_save,        encoding='utf_8_sig',
index=False)


def ra_break():
    x = np.arange(0,1, 0.01)
    y = exp(-3*x)-0.2
    plt.plot(x,y, label='y=exp(-3x)-0.2')
    plt.legend()
    plt.grid(True)
    plt.xlabel('企业“可靠性”', fontsize=12.5)
    plt.ylabel('企业违约率', fontsize=12.5)
    plt.show()


def degree_eff():
    x = np.arange(-3, 3, 0.1)
```

```
    y_1 = -1/(1+exp(-2*x))+1
    y_2 = -1/(1+exp(-x))+1
    y_3 = -1/(1+exp(-0.5*x))+1
    plt.plot(x, y_1, label='y=-1/(1+exp(-2*x))+1', color="red")
    plt.plot(x, y_2, label='y=-1/(1+exp(-x))+1', color="blue")
    plt.plot(x, y_3, label='y=-1/(1+exp(-0.5*x))+1', color="green")
    plt.legend()
    plt.grid(True)
    plt.xlabel('企业规模', fontsize=12.5)
    plt.ylabel('企业受影响程度', fontsize=12.5)
    plt.show()


def change(money, total_money, raw_data, num):
    reed = [random.random() for i in range(num+1)]

    total = sum(reed)
    count_add = 0
    count_de = 0
    flag_add = 0
    flag_de = 0
    for i in range(len(raw_data)):
        if flag_add == 0:
            if  raw_data.iloc[i]['信誉评级']  ==  'A'  or
raw_data.iloc[i]['信誉评级'] == 'B':
                money[i] = min(money[i] + reed[count_add] / total *
total_money, 100)
                count_add += 1
                if count_add == num:
                    flag_add = 1
        if flag_de == 0:
            if raw_data.iloc[i]['信誉评级'] == 'C':
                count_de += 1
                money[i] = max(money[i] - reed[count_de] / total *
total_money, 10)
                if count_de == num:
                    flag_de = 1
        if flag_add == flag_de == 1:
            break
    return money


def strategy_adj(path_raw, path_save):
    raw_data    =    pd.read_csv(path_raw,    encoding='utf_8_sig',
```

```
engine='python')
    rate = list(raw_data['借贷年利率'])
    for i in range(len(rate)):
        if rate[i]>0:
            rate[i] = max(rate[i]-random.random()*3, 4)
    money = list(raw_data['借贷金额（万元）'])
    money_1 = change(money, 4000, raw_data, num=80)
    money_2 = []
    money_3 = []
    for i in money_1:
        if i!= 0:
            money_2.append(round(max(min(-
5+10*random.random()+i,100),10),2))
        else:
            money_2.append(0)
    for i in money_1:
        if i!= 0:
            money_3.append(round(max(min(-
10+20*random.random()+i,100),10),2))
        else:
            money_3.append(0)

    # money_2 = change(money, 5000, raw_data, num=50)
    # money_3 = change(money, 8000, raw_data, num=80)
    print(len(money_3))
    pd.DataFrame({'企业代号': list(raw_data['企业代号']), '信誉评级':
list(raw_data['信誉评级']),
                  '借贷金额（万元，0.5）': money_1, '借贷金额（万元，1）
': money_2, '借贷金额（万元，2）': money_3,
                  '借贷年利率（%）':rate}).to_csv(path_save,
encoding='utf_8_sig', index=False)



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'
    path_index = r'../../B/result/index_2.csv'
    path_score = r'../../B/result/score.csv'
    path_pre = r'../../B/result/附件二企业信誉评级识别结果.csv'
    # strategy_money(path_pre, r'../../B/result/问题二借贷方案.csv')
    # ra_break()
    # degree_eff()
```

```
    # pre_analysis(path_pre)

    strategy_adj(r'../../B/result/ 问 题 二 借 贷 方 案 .csv',
r'../../B/result/借贷方案调整策略.csv')
    # loss_fig(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')
    # plot_score(path_score)
    #                                  comprehensive_score(path_score,
path_save=r'../../B/result/score_all.csv')

    # score_comp(path_index, path_save=r'../../B/result/score_1.csv')
    # correlation(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)

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

8.2 深度神经网络模型

```
import numpy as np
import pandas as pd
import torch
import itertools
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from   sklearn.metrics   import   accuracy_score,   confusion_matrix,
recall_score, f1_score, roc_curve, auc
import torch.utils.data
import matplotlib.pyplot as plt
import math
from sklearn.manifold import TSNE
# from pylab import
import random

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'

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

```
def data_prepare(data_raw):
    df_norm = (data_raw - data_raw.mean()) / (data_raw.std())
    df_norm[0:110].to_csv(r'../../../train_norm.csv',
encoding='utf_8_sig', index=False)
    df_norm[110:].to_csv(r'../../../test_norm.csv',
encoding='utf_8_sig', index=False)
    #                  df_norm[120:].to_csv(r'../../../val_norm.csv',
encoding='utf_8_sig', index=False)


class nn_embedding(nn.Module):
    def __init__(self, input_size, class_num):
        super(nn_embedding, self).__init__()
        # self.embedding = nn.Embedding(attributes_num, dim_embedding)
        self.linear1 = nn.Linear(input_size, 64)
        self.dropout1 = nn.Dropout(0.5)
        self.linear2 = nn.Linear(64, 64)
        # self.linear3 = nn.Linear(128, 256)
        # self.linear4 = nn.Linear(256, 512)
        self.linear5 = nn.Linear(64, class_num)
    #
    def forward(self, inputs):
        # embeds = self.embedding(inputs).view(1, -1)
        #    embeds    =    torch.cat((embeds,    torch.tensor([[1]],
dtype=torch.float)), 1)

        out_1 = F.tanh(self.linear1(inputs))
        # out_2 = F.tanh(self.linear2(out_1))
        # out_3 = F.tanh(self.dropout1(self.linear3(out_2)))
        # out_4 = F.tanh(self.dropout1(self.linear4(out_3)))
        out = F.tanh(self.linear5(out_1))
        log_probs = F.log_softmax(out)
        return log_probs


def loss_iteration():
    # print([np.round(i,4) for i in loss_train])
    # print([np.round(i,4) for i in loss_val])
    # print([np.round(i,4) for i in F1_train])
    # print([np.round(i,4) for i in F1_val])
    loss_train = [1.5166, 1.3879, 1.2181, 1.1747, 1.1467, 1.0282,
1.1157, 1.0069, 1.0007, 0.9961, 0.9927, 0.9901, 0.9881, 0.9865,
```

```
    0.9852, 0.984, 0.9831, 0.9823, 0.9816, 0.981, 0.980, 0.973, 0.970,
0.970, 0.968]
    loss_val = [1.8195, 1.6394, 1.389, 1.5563, 1.2349, 1.1206, 1.0108,
1.0039, 0.999, 0.9953, 0.9925, 0.9904, 0.9887, 0.9873,
    0.9862, 0.9853, 0.9845, 0.9838, 0.9833, 0.9827, 0.9820, 0.9814,
0.9810, 0.9771, 0.9770]
    F1_train = [0.4008, 0.5475, 0.6135, 0.6722, 0.716, 0.7484, 0.7784,
0.8162, 0.8104, 0.8114, 0.821, 0.8292, 0.829, 0.8221,
    0.8401, 0.8234, 0.8136, 0.8328, 0.8333, 0.8201, 0.8322, 0.8334,
0.8200, 0.8212, 0.832]
    F1_val = [0.5357, 0.6128, 0.6522, 0.6874, 0.7321, 0.7772, 0.7976,
0.7417, 0.7891, 0.8081, 0.8004, 0.8012, 0.802, 0.8161,
    0.8165, 0.8133, 0.8134, 0.8231, 0.809, 0.8114, 0.8113, 0.8134,
0.8131, 0.8292, 0.828]


    iteration = [i for i in range(len(loss_train))]
    plt.plot(iteration,   loss_train,   color='red',   linestyle='-',
label='training loss')
    plt.plot(iteration,   F1_train,   color='red',   linestyle='-.',
label='training F1')
    plt.plot(iteration,   loss_val,   color='green',   linestyle='-',
label='validation loss')
    plt.plot(iteration,   F1_val,   color='green',   linestyle='-.',
label='validation F1')
    plt.xlabel('iteration')
    plt.ylabel('loss & F1')
    plt.legend()
    plt.grid(True)
    #       plt.savefig(r'../results/classification/nn_training.png',
dpi=300)
    plt.show()


def training(data_train, label_train, data_test, label_test):
    losses_train = []
    losses_test = []
    train_num = len(data_train)
    loss_function = nn.NLLLoss()  # negative log likelihood combine
with Softmax = cross entropyJoint learning
    model = nn_embedding(13, 4)
    model.to(device)
    optimizer     =     optim.SGD(model.parameters(),     lr=0.001,
weight_decay=0.1)
```

```
    #    scheduler    =    torch.optim.lr_scheduler.StepLR(optimizer,
step_size=20, gamma=0.8)
    iteration_num = 50
    test_F1 = []
    train_F1 = []
    best_f1 = 0
    #
model.load_state_dict(torch.load('../results/classification/model_nod
ataenhancement_embedding100_line256_iteration20.pkl'))
    for epoch in range(iteration_num):
        total_loss = 0
        predict_list_temp = []
        for i in range(len(data_train)):
            # Step 1. Prepare the inputs to be passed to the model
            train_sample      =      torch.tensor(data_train.iloc[i],
dtype=torch.float).to(device)
            # Step 2. Recall that torch *accumulates* gradients. Before
passing in a new instance, you need to zero out
            # the gradients from the old instance
            model.zero_grad()
            # Step 3. Run the forward pass, getting log probabilities
of classes
            log_probs = model(train_sample)
            _, predict = torch.max(log_probs.data, 0)

            predict_list_temp.append(predict.to('cpu'))
            # Step 4. Compute your loss function. (Again, Torch wants
the target class wrapped in a tensor)
            loss       =       loss_function(log_probs.unsqueeze(0),
torch.tensor([label_train[i]], dtype=torch.long).to(device))

            # Step 5. Do the backward pass and update the gradient
            loss.backward()
            optimizer.step()

            # scheduler.step()
            # Get the Python number from a 1-element Tensor by calling
tensor.item()
            total_loss += loss.item()

        with torch.no_grad():
            test_temp = []
            test_loss = 0
            test_num = len(data_test)
```

```
            for k in range(len(data_test)):
                test_sample     =     torch.tensor(data_test.iloc[k],
dtype=torch.float).to(device)
                log_probs = model(test_sample)
                _, predict = torch.max(log_probs.data, 0)

                test_temp.append(predict.to('cpu'))
                loss     =     loss_function(log_probs.unsqueeze(0),
torch.tensor([label_test[k]], dtype=torch.long).to(device))
                test_loss += loss.item()
            losses_test.append(test_loss / test_num)
            test_F1.append(f1_score(label_test,             test_temp,
average='weighted'))
            print("Iteration: %f" % epoch)
            print("test accuracy: %.4f" % accuracy_score(label_test,
test_temp))
            print("test  recall:  %.4f"  %  recall_score(label_test,
test_temp, average='weighted'))
            print("test F1: %.4f" % f1_score(label_test, test_temp,
average='weighted'))

        if  f1_score(label_test,  test_temp,  average='weighted')  >
best_f1:
            best_f1      =      f1_score(label_test,      test_temp,
average='weighted')
            best_model = model

        losses_train.append(total_loss / train_num)
        print("Train     F1:     %.4f"     %     f1_score(label_train,
predict_list_temp, average='weighted'))

        train_F1.append(f1_score(label_train,      predict_list_temp,
average='weighted'))
    # print(losses_train)  # The loss decreased every iteration over
the training data!
    # loss_iteration(losses_train, losses_test, train_F1, test_F1)

    #                                 torch.save(model.state_dict(),
'../results/classification/model_nodataenhancement_embedding50_line64
_iteration20.pkl')
    return best_model


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 roc_plot(data_test, best_model, label_test):
    num_test = len(data_test)
    predict_list = []
    predict_prob = []
    best_model.to('cpu')
    for k in range(num_test):
        test_sample = torch.tensor(data_test.iloc[k],
dtype=torch.float)
        output_predict = best_model(test_sample)
        pred, predict = torch.max(output_predict.data.unsqueeze(0), 1)
        predict_prob.append(output_predict.data)
        predict_list.append(predict)

    print("Test accuracy: %.4f" % accuracy_score(label_test,
predict_list))
    print("Test recall: %.4f" % recall_score(label_test, predict_list,
average='weighted'))
    print("Test F1: %.4f" % f1_score(label_test, predict_list,
average='weighted'))
    one_hot = []
    for i in label_test:
        temp = [0] * 4
        temp[i] = 1
        one_hot += temp
    print(predict_prob)
    fpr, tpr, _ = roc_curve(one_hot,
list(itertools.chain.from_iterable(predict_prob)))
    ROC_curve(fpr, tpr, auc(fpr, tpr))
```

```
def main():
    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 = ['销售规模（对数）', '销项开票数量（对数）', '平均销
项开票金额（对数）', '进购规模（对数）', '进项开票数量（对数）',
                  '税价比', '发票平均税率', '下游客户数量', '指定周期
负数发票金额比率', '指定周期作废发票金额比率', '固定周期下游客户数量',
                  '销方基尼系数', '销方 HIHI 指数']
    loss_iteration()
    #   train_data_norm   =   pd.read_csv(r'../../../train_norm.csv',
encoding='utf_8_sig')
    #    test_data_norm    =    pd.read_csv(r'../../../test_norm.csv',
encoding='utf_8_sig')
    #  val_data  =  pd.read_excel(r'../../problem   C   附件 2.xlsx',
encoding='utf_8_sig')
    # train_sp = train_data[index_temp]
    # test_sp = test_data[index_temp]
    # val_sp = val_data[index_temp]
    # val_sp.columns = [str(i) for i in range(1, 21)]
    # all_data = pd.concat([train_sp, test_sp])
    # data_prepare(all_data)
    # train_label = label_creat(list(train_data['信誉评级']))
    # test_label = label_creat(list(test_data['信誉评级']))

    #    best_model    =    training(train_data_norm,    train_label,
test_data_norm, test_label)
    # roc_plot(test_data_norm, best_model, test_label)
    # loss_iteration()


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