中小微企业信贷风险量化与冲击响应分析
——基于时间序列和机器学习

摘要

中小微企业规模较小，贷款风险大，因此银行需要依据企业的多个方面对其进行信用评估，然后向信誉好风险低的企业发放不同额度的贷款。

本文的主要目标是根据以往有限的企业交易数据等，分析企业经营情况对企业信用的影响，从而制定合理的放贷策略，帮助银行规避风险，尽可能扩大银行贷款利润。

在建立模型前，我们对数据进行了多方位的观察和分析，同时进行了数据预处理和清洗，剔除了对我们建模帮助不大的部分数据。

对于**问题一**，我们试图利用已有数据，来分析银行给企业评定信用等级的方法。第一步用数据挖掘的手段，从企业发票信息中得到**企业净值月中位数序列**，并用时间序列分析的知识确定了企业未来几个月的**偿还能力**。第二步，我们从发票信息中提取有关指标，分析最重要的几个指标，利用机器学习中的 **AdaBoost** 分类模型建立一个**企业经营-信用评估模型**，将银行向企业贷款的风险量化。最后综合多种约束条件，通过考虑银行获得的期望收益，建立了一个**最大化期望收益优化模型**，从而帮助银行合理制定贷款方案。

对于**问题二**，首先用与问题一相似的方式获得企业净值月中位数序列，对企业的偿还能力进行预测。第二步从附件二的企业发票信息中提取问题一中用到的指标，然后在问题一建立的模型的基础上，应用企业信用等级评估模型，对附件二给出的 302 家企业进行**信用评估**，从而确定它们的信用等级。确定完这些企业的信用等级后，我们可以再次利用问题一的**优化模型**，给出银行的贷款方案。

对于**问题三**，我们考虑突发事件对各行业的不同影响，选取各行业的股票指数建立**向量自回归（VAR）模型**。依次通过**差分平稳性检验**，**Johansen 协整性检验**，**滞后期选择过程**，我们得到突发事件下各行业的**脉冲响应图**，定量刻画突发事件的冲击。最后将脉冲信号作用于企业净值序列上，使用问题二的信用评估得出信贷策略。我们通过新冠疫情事件梳理模型具体建立过程，其预测趋势与行业增加值，冲击深度等指标符合较好；同时可应用于次贷危机分析，体现模型普适性。

综上，本文从**数据挖掘与数据分析**的角度入手，利用多种手段，分别建立了企业信用等级评估模型和银行最大化期望收益优化模型，同时还对突发公共事件的影响进行了量化。观察模型给出的策略，较为符合我们的日常认知，可见模型有一定的可用性，可以为银行提供一定程度的指导。

关键字：数据挖掘　信用评级　最大化收益　中小微企业

一、 问题重述

中小微企业是国民经济的毛细血管，作为最基层、最贴近人民群众的经济组织，中小微企业在刺激消费需求，提供就业岗位，拉动经济增长等诸多方面发挥着不可忽略的作用。中小微企业向银行进行贷款申请时，面临着大企业和其他小微企业的竞争，由于企业体量小、固定资产和可抵押物相对偏少，常常处于劣势。小微企业贷款融资难的局面，不仅陷中小微企业于贷款困境之中，同时也会让商业银行损失一笔潜在的利息收入，因此，商业银行有必要去探索针对中小微企业的信用评定和风险评估机制，合理设置贷款额度和贷款利率，以追求商业银行和中小微企业的双赢局面。

本题由三个小问组成。问题一要求我们定义和描述一种风险评估的方法或指标，跟据已知的企业的信用评级以及是否有违约记录，评估题目中的 123 家企业的风险大小，并将银行所具有的贷款额度按照预期收益最大、风险最小等原则进行组合优化，确定每一家企业可以获得的贷款份额。

问题二则是给出了 302 家没有信贷记录的相关企业，已知这些企业在 2017-2020 三年左右的时间内的进项和销项发票信息，需要我们进行信用等级评定和风险评估，从而确定一亿元的贷款划分方法。

问题三要求进一步考虑企业在贷款经营期间可能遇到的突发因素，如新冠肺炎疫情、全球经济危机等黑天鹅事件的影响下企业盈利能力和还款能力的变化，并由此调整在问题二之中给出的贷款分配方案，使之具有更强的鲁棒性。

二、 问题分析

2.1 问题一的分析

问题一的要求是在企业交易数据、信用评级和违约历史信息相对充足的情况下为银行建立一个**风险评估模型**，然后结合各种约束条件建立一个**优化模型**，考虑如何最大化银行的期望收入。

结合相关资料，我们了解到银行在决定是否向企业提供贷款前，会从多个角度调查企业的经营状况，对企业的信贷风险进行评估。通过观察，我们发现附件一给出的企业的数据较为完整，有企业的信用等级以及企业是否发生违约的相关信息。所以我们认为应该从数据入手，挖掘出企业信用等级与其经营状况的关系。从企业的进销项发票信息中，尤其是从每一次有效的交易发票的价税合计及其累加值中观察企业的交易对象数、交易额分布，并从销方单位代号、购方单位代号等信息发掘出企业的上下游供求关系稳定程度等，进而量化企业的信用风险，建立一个**企业经营-信用评估模型**。

评估完企业的信贷风险后，我们还需要设计贷款策略。经过我们的讨论分析，可以尝试建立一个**概率模型**，来量化企业面对不同贷款额度、利率的情况下的决策以及后续的履约情况给银行带来的收益，从而得到一个完整的**最大化期望**的高维多变量优化模型。然后编写代码得到此种情况下银行的最佳贷款策略。

2.2 问题二的分析

问题二的总体要求和问题一类似，还是要求我们建立一个风险评估模型，提出银行的贷款策略。主要差别在于问题一中的 123 家企业都是具有贷款记录的企业，因此银行具有他们的信用等级和违约记录等信息，而问题二之中，302 家企业的这些信息都是未知的。

所以，我们可以利用问题一中建立起来的**企业经营-信用评估模型**对附件二中的无信用记录的企业作出信用等级评估，并依据各个等级的违约率，然后再次利用问题一的**最大化期望优化模型**设计一个银行的贷款策略。

2.3 问题三的分析

问题三引入了外部冲击，要求在前两问模型基础上，考虑突发事件对企业风险和贷款方案的影响。由于各行业受冲击程度有所差别，我们选取各行业股价作为时间序列向量，视为内生变量，建立向量自回归（VAR）模型。模型通过可行性检验后，我们将突发事件视为脉冲，通过 VAR 模型的脉冲响应分析突发事件对各行业的冲击程度。将脉冲信号按行业一致地作用在企业资金净值数据上，得出量化分析信贷风险与策略。

我们以新冠肺炎疫情作为主要事例分析 VAR 模型的建立应用过程，并根据突发事件时期的行业增加值和冲击深度指标，实际验证模型的趋势。同样在 2008 年金融危机事件中应用 VAR 模型，方法具有较高的普适性。

三、 模型假设与符号说明

1. 企业的经营业务相对稳定，短期内不会出现产业转型、产权转移等大规模变动；
2. 对于相同信用等级的企业，银行的客户损失率只与向该企业提供的贷款利率有关；
3. 银行总是以追求最大利益为经营目标，尽量将贷款的预算用尽；
4. 企业有一定概率放弃在本银行贷款，而且企业只会由于利率过高而放弃贷款；
5. 企业获取到贷款后有概率违约，若某一家企业发生违约，则银行完全损失贷款本金；
6. 一家企业是否违约仅由其自身经营状况决定，不同企业之间的违约与否是独立的；
7. 突发事件对同一行业中的不同企业影响相近，对不同行业之间的影响可能不同。

下面的表 1是我们建模中用到的主要符号以及说明。

表 **1**　符号说明

符号	意义	单位	备注
R_i	企业 i 的信用等级	——	取 A,B,C,D 之一
$F_{i,j}$	企业 i 的第 j 个信用评估因子	——	$j = 1, 2, 3, 4, 5$
$\mathbf{F}$	企业信用评估因子矩阵	——	$\mathbf{F} = (F_{i,j})$
$\bar{p}_i(R_i)$	企业 i 得到贷款后违约的概率	——	$0 \leq p_i \leq 1$
$p_i(R_i)$	企业 i 得到贷款后履约的概率	——	$0 \leq p_i \leq 1$
D_i	企业 i 的最大偿还能力	元	$10^5 \leq D_i \leq 10^6$
$\mathbf{D}$	D_i 构成的列向量	——	$\mathbf{D} = (D_1, D_2, \cdots)^T$
A_i	银行决定给企业 i 的实际贷款	元	0 或 $10^5 \leq A_i \leq D_i$
$\mathbf{A}$	A_i 构成的列向量	——	$\mathbf{A} = (A_1, A_2, \cdots)^T$
I_i	银行决定给企业 i 的贷款利率	——	$0.04 \leq I_i \leq 0.15$
$\mathbf{I}$	I_i 构成的列向量	——	$\mathbf{I} = (I_1, I_2, \cdots)^T$
$g(I_i, R_i)$	企业 i 放弃贷款的概率	——	——
$f(I_i, R_i)$	企业 i **不**放弃贷款的概率	——	$f(I_i, R_i) = 1 - g(I_i, R_i)$
X_i	银行从企业 i 得到的收益	元	随机变量
M	银行能提供的最大贷款总量	元	$\sum_i A_i \leq M$
Q	银行获得的期望收益	元	——

四、 数据处理与模型准备

在做数据处理之前需要我们对数据进行**清洗**，剔除一些无用的数据。首先，我们认为作废发票对我们的分析不起作用，因此将所有的作废发票数据剔除；其次，根据题目的提示，我们记录和统计所有的负数发票的价税合计总额，便于后续处理；除此之外，检验题目给出的数据之中是否有明显的异常值。具体代码如 [**附录一**] 所示。

五、 模型建立与求解

5.1 问题一的求解

5.1.1 建立风险评估模型

为了建立风险评估模型，我们研究 123 家企业的销项发票信息和进项发票信息，从中提取得到与企业还款能力和经营风险直接相关的五个因子，进行定量评估：

首先，无论是进项发票还是销项发票，这些具有交易日期的发票都可以进一步处理称为时间序列信息，为了对企业未来六个月到一年内的经营利润进行预测和估计，我们进行了时间序列分析。

对于 123 家企业之中的每一家企业，从最早具有交易记录的日期开始，到最晚有交易记录的日期为止，以 0 作为公司净值的初始值，以月为时间单位，运行 [**附录二**] 计算由于购买原材料和出售货物所导致的公司净值的变化，从而得到净值时间序列。

选择评级分别是 A，B，C，D 的企业，运行 [**附录三**] 对其净值时间序列进行绘图，如图 1所示。

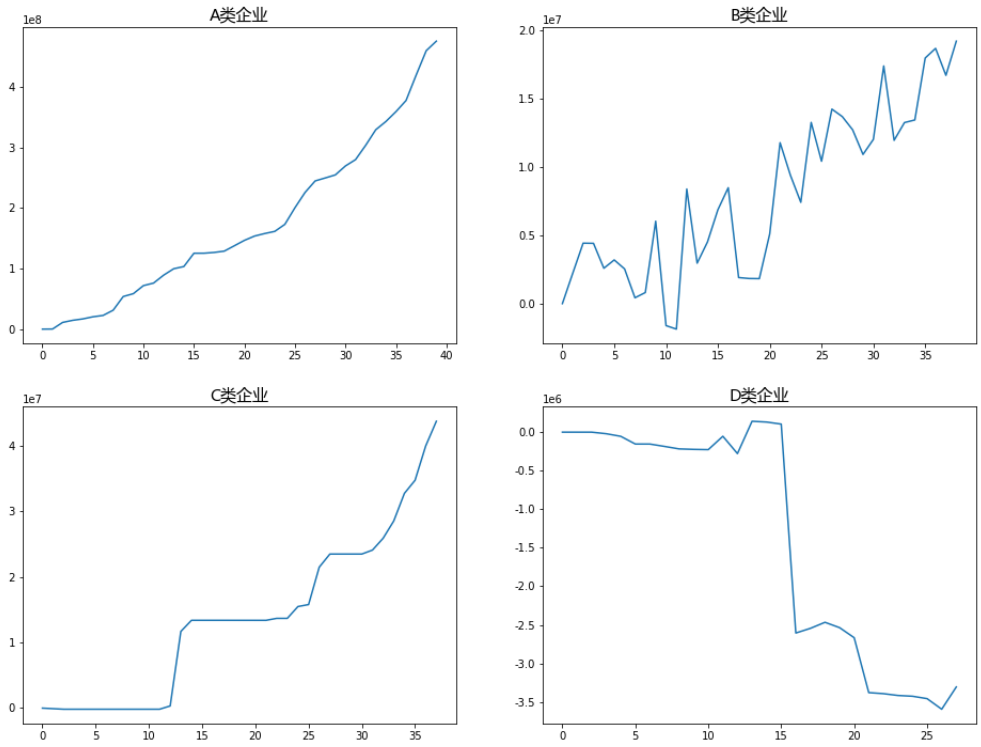

图 **1**　各类企业净值变化

对于银行而言，企业未来偿债能力相对于过去的业绩更值得考虑。从图中可以大

致推断企业净值是一阶差分平稳的，因此我们决定使用 ARIMA 模型进行时间序列的预测。ARIMA 基于自回归移动平均模型（ARMA），序列差分后，满足 [1]

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{1}$$

再次运行 [**附录三**] 对时间序列一阶差分后，经检验序列平稳性整体良好，部分 ACF 图如图 2所示：

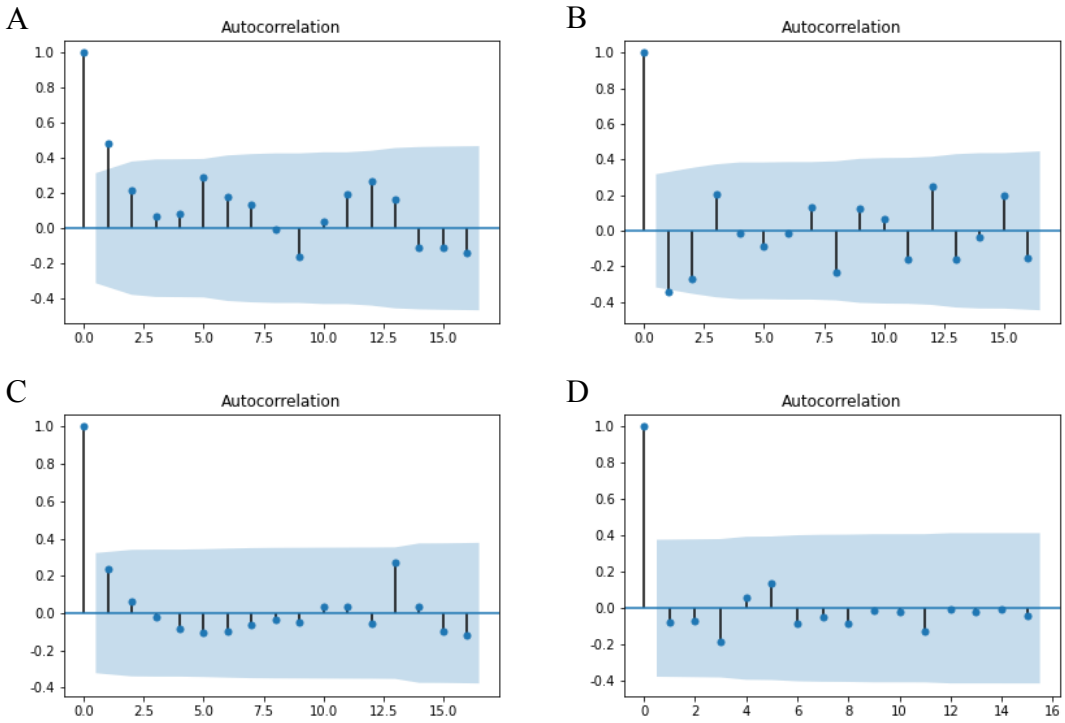

图 **2**　各类型企业 **ACF** 图

使用 Python 中 acorr_ljungbox 进行白噪声检验，返回 p 值整体位于 0.05 以下，因此可通过 statsmodels.tsa 中的 arima_model 进行时间序列预测。为保证精准度，我们通过编写 [**附录四**] 的 Python 程序，利用 ARIMA 得出每一家企业在 2020 年 1 至 6 月份的净值预测值，由于净值的大小反映了一家企业的体量以及资金流动能力，取其六个月数据的中位数作为评估风险的关键性因子之一。

同时，对于一家平稳经营的企业，其公司净值应当趋于稳定，短时间之内不应有大的波动。所以我们计算每一家企业在这六个月之中的净值平均值和标准差，以其变异系数作为我们评价其经营平稳性和贷款风险大小的第二个重要指标。

为了研究每一家企业的供应销售链条的稳定性，从销方发票信息和买方发票信息之中找出上下游企业，对所有交易的数额求偏度，以衡量该企业对于大额交易和小额交易的倾向性；将该企业所进行交易的每一家上下游企业的交易额进行汇总统计，计算其峰度，用于衡量该企业对于稳定供货商的倾向性。

此外，我们认为，如果一家企业的销项发票信息之中有过多的负数发票，代表着这家企业的货物质量和企业信誉可能存在一定的问题，有一定概率遭遇信任危机。因此，我们统计了 123 家企业的负数发票的价税合计金额在销项发票价税合计总额之中的比重，作为最后一个量化因子。

综上所述，我们得到的五个决策因子分别是：

- $F_{i,1}$ 第 i 家企业由时间序列分析得到的未来六个月公司净值的中位数；
- $F_{i,2}$ 第 i 家企业由时间序列分析得到的未来六个月公司的净值变异系数；
- $F_{i,3}$ 第 i 家企业所有交易发票的价税合计金额的偏度值；
- $F_{i,4}$ 第 i 家企业所有交易对象的交易金额的峰度值；
- $F_{i,5}$ 第 i 家企业所有交易发票之中负数发票的金额占交易发票总金额的比重。

运行 [**附录五**] 获得企业决策因子后，我们使用机器学习中的分类模型进行企业风险评估准确率的观察。

首先使用支持向量机（SVM）对上述六个风险量化因子与附件一中给出的企业信用评级之间的关系进行拟合探索。SVM 的工作原理是把每一个数据看作高维空间中的一个点，将特性记为 $\mathbf{x}_i$，标签记为 y_i，然后寻找一个超平面 $y =< \mathbf{w}, \mathbf{x} > +b$，使得各点到该分类平面的距离尽可能大，即：

$$\begin{aligned} &\min_{\mathbf{w},b} |\mathbf{w}|^2, \\ s.t. \quad &y_i(< \mathbf{w}, \mathbf{x} > +b) \geq 1, (i = 1, 2, \cdots, N) \end{aligned} \tag{2}$$

若数据并非完全线性可分，则可引入惩罚因子 C, 从而建立软边距的 SVM 模型；又或是使用核函数 $\Psi(\mathbf{x})$ 将其映射到更高维度的空间上，使其变为线性可分的 [2]。

将 123 家企业的因子以及评级，按照 7:3 的比例划分为训练集和测试集，经过 SVM 训练得到分类器，其准确率为 0.4054。

为了进一步提升训练和分类效果，得到性能更优秀的分类器，我们同时尝试使用机器学习中的 AdaBoost 模型进行训练。AdaBoost 的运行原理主要是通过线性组合多个弱分类器而构成一个强分类器 [3]，即 $h_i(\mathbf{x})$ 是数个较弱的分类器，通过算法步骤构建一个强分类器

$$f(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x}) \tag{3}$$

其中各 α_i 是算法得到的弱分类器的权重。

运行 [**附录六**] 中的代码，将 123 家企业的因子以及评级，按照 7:3 的比例划分为训练集和测试集，AdaBoost 分类器最终的分类效果为 0.6852。

上述两种模型的预测准确率如表 2所示：

表 **2**　各分类模型的预测准确率

模型	**SVM**	**AdaBoost**
准确率	0.4054	0.6852

经过比较，我们可以选择准确率较高的 AdaBoost 分类模型作为我们的风险评估模型，用它来对企业的信用等级进行评估。

5.1.2 建立最大化期望优化模型

银行的基本目标就是最大化贷款收益，所以我们要量化银行的收益期望。首先我们要分析银行从每个企业的贷款中能获得的收益，主要是三种情况：

1. 企业获得贷款，并且履约。这种情况下，银行能获得利润 A_iI_i，此情况发生的条件是客户没有流失，并且企业履约，概率是 $f(I_i, R_i)p_i(R_i)$；
2. 企业因为利率过高而放弃贷款，即银行损失了客户。这种情况下，银行不能获得利润，此情况发生的概率是 $1 - f(I_i, R_i)$；
3. 企业获得贷款，但没有履约。这种情况下，银行损失本金 A_i，此情况发生的条件是客户没有流失，但企业不履约，概率是 $f(I_i, R_i)(1 - p(R_i))$。

所以可以将银行能从一家企业获得的利润 X_i 视为一个随机变量，它的概率分布如表 3所示：

表 **3**　X_i 的概率分布

X_i	$-A_i$	0	A_iI_i
P	$f(I_i, R_i) \cdot (1 - p_i(R_i))$	$1 - f(I_i, R_i)$	$f(I_i, R_i) \cdot p_i(R_i)$

接下来，为了测算各信用等级企业的违约率，我们编写程序，统计了附件一中企业的违约情况和对应的等级，将各等级企业违约的频率视作概率。运行 [**附录七**] 中的程序，得到下面的图 3：

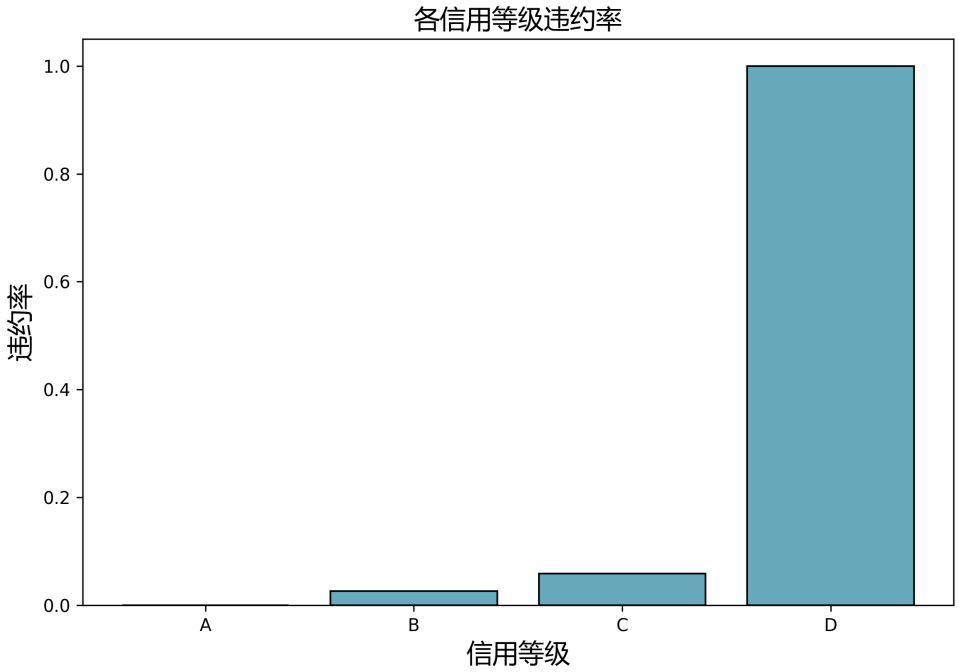

图 **3**　信用等级-违约率

具体统计数据如下所示：

$$\bar{p}_i(R_i) = \begin{cases} 0.0000, & R_i = A, \\ 0.0263, & R_i = B, \\ 0.0588, & R_i = C, \\ 1.0000, & R_i = D. \end{cases} \tag{4}$$

由于我们统计的是各等级企业违约的概率，而企业履约是其对立事件，所以概率为：

$$p_i(R_i) = 1 - \bar{p}_i(R_i) = \begin{cases} 1.0000, & R_i = A, \\ 0.9737, & R_i = B, \\ 0.9412, & R_i = C, \\ 0.0000, & R_i = D. \end{cases} \tag{5}$$

由于风险等级为 D 类的企业违约率高，银行贷款风险过大，所以原则上银行不会给评估风险为 D 级的企业发放贷款，所以在考虑银行的贷款策略时，我们不会考虑 D 类企业，也即 D 类企业的 $A_i = 0$。

之后，我们还需要分析在不同利率水平下，不同信用等级的企业不放弃贷款的概率 $f(I_i, R_i)$。此处我们利用了附件三给出的数据，使用 Matlab 的工具箱拟合出不同利率水平下，不同信用等级的企业的流失率，即放弃贷款的概率 $g(I_i, R_i)$：

$$g(I_i,R_i)=\begin{cases}-76.41I_i^2+21.98I_i-0.6971, & R_i=A, \quad (R^2=0.993, SSE=0.014)\\ -67.93I_i^2+20.21I_i-0.6504, & R_i=B, \quad (R^2=0.9945, SSE=0.01037)\\ -63.94I_i^2+19.57I_i-0.6393, & R_i=C, \quad (R^2=0.9951, SSE=0.0094).\end{cases} \tag{6}$$

运行 [**附录八**]，得到散点图与回归函数图如图 4所示：

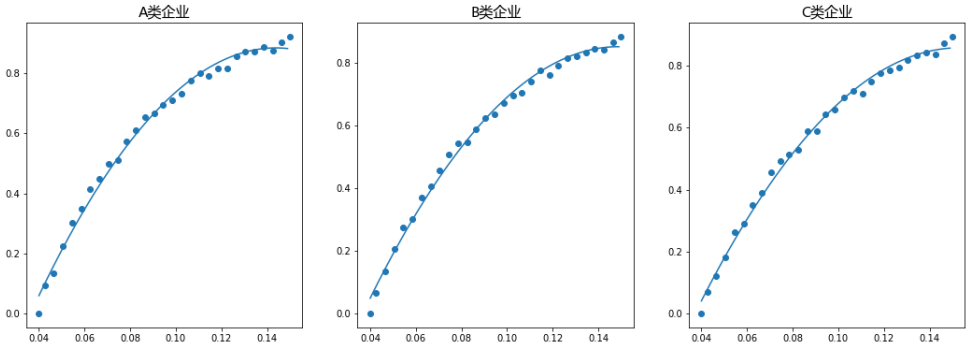

图 **4**　各信用等级在不同利率水平下的流失率

通过 f 与 g 的关系我们可以得到

$$f(I_i,R_i)=1-g(I_i,R_i)=\begin{cases}76.41I_i^2-21.98I_i+1.6971, & R_i=A,\\ 67.93I_i^2-20.21I_i+1.6504, & R_i=B,\\ 63.94I_i^2-19.57I_i+1.6393, & R_i=C.\end{cases} \tag{7}$$

有了上述公式后，我们便可以量化银行能从一家企业得到的期望利润 $\mathbb{E}(X_i)$ 了：

$$\begin{aligned}\mathbb{E}(X_i)&=-A_i\cdot f(I_i,R_i)\cdot(1-p_i(R_i))+A_iI_if(I_i,R_i)\cdot p_i(R_i)\\&=-A_if+A_ip_if+I_iA_ip_if\\&=A_if(I_i,R_i)\cdot(-1+p_i(R_i)+I_ip_i(R_i))\\&=\begin{cases}A_iI_i(76.41I_i^2-21.98I_i+1.6971), & if \quad R_i=A,\\ A_i(67.93I_i^2-20.21I_i+1.6504)(-0.0263+0.9737I_i), & if \quad R_i=B,\\ A_i(63.94I_i^2-19.57I_i+1.6393)(-0.0588+0.9412I_i), & if \quad R_i=C.\end{cases}\end{aligned} \tag{8}$$

最后我们回归到银行的基本目标——最大化期望利润，可以写出银行总利润的表达式

$$Q = \sum_{i=1}^{n} \mathbb{E}(X_i) \tag{9}$$

优化目标是

$$\begin{aligned} &\max Q = \sum_{i=1}^{n} \mathbb{E}(X_i) \\ &s.t. \begin{cases} 10 \le D_i \le 100, \\ A_i = 0, \text{或} \quad 10 \le A_i \le D_i, \\ 0.04 \le I_i \le 0.15, \\ \sum\limits_i A_i \le M. \end{cases} \end{aligned} \tag{10}$$

由于我们希望使用 Matlab 的 fmincon 函数，该函数是寻求目标函数的最小值，所以需要将优化目标改写为原来的银行总利润的相反数。结合实际情况和题意，可以写出优化目标以及约束条件如下所示：

$$\begin{aligned} &\min -Q = -\sum_{i=1}^{n} \mathbb{E}(X_i) \\ &s.t. \begin{cases} 10 \le D_i \le 100, \\ A_i = 0, \text{或} \quad 10 \le A_i \le D_i, \\ 0.04 \le I_i \le 0.15, \\ \sum\limits_i A_i \le M. \end{cases} \end{aligned} \tag{11}$$

此处我们还需要量化一个企业能够偿还的最大债务，我们决定使用上面得到的企业 6 个月后的净利润。现在，除贷款总额 M 之外，我们已经能够确定上面的优化式子的全部变量值。我们编写了 Matlab 程序，对于附件一给出的 123 家企业，给出一个确定的贷款总额，运行 [**附录九**] 及 [**附录十**]，即可得到一个银行的贷款分配策略。表 4是以 1 亿元为例运行程序得到的示例结果（部分，详见支撑材料中的 [数据/schme1.xlsx]）：

表 4　问题一的贷款策略（部分）

企业代号	贷款金额（元）	贷款利率
E1	100000.0	0.06162
E2	965935.0	0.06236
E3	939198.1	0.15000
……	……	……
E121	0.0	0.13776
E122	0.0	0.04023
E123	0.0	0.10935

5.2 问题二的求解

附件二给出了 302 家没有信贷记录，只有企业发票记录，所以我们要先对企业的信用等级进行评估。与问题相似，我们还是要清洗发票数据。运行 [**附录十一**] 至 [**附录十四**] 中的程序，得到提取出的指标并将其送入问题一中训练好的企业经营-信用模型进行预测，获取这些企业的信用评级。[**附录十五**] 统计了预测得到的企业信用评级分布，如图 5所示：

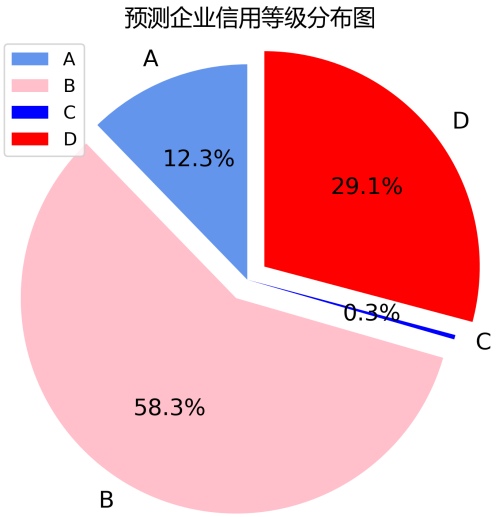

图 5　预测企业信用评级分布

有了企业的信用评级后，将这些数据送入题目一的最大化期望收益优化模型，运行 [**附录十六**] 和 [**附录十七**] 得到最终银行的贷款分配策略。部分结果如下所示（详见支撑材料中的 [数据/schme2.xlsx]）：

表 **5**　问题二的贷款策略（部分）

企业代号	贷款金额（元）	贷款利率
E124	100000.0	0.07875
E125	749133.5	0.08030
E126	749133.5	0.04006
……	……	……
E423	0.0	0.09899
E424	0.0	0.09899
E425	100000	0.07413

5.3 问题三的求解

问题三则要求我们考虑突发公共事件对企业信用的影响和对银行贷款策略的影响。突发事件可影响企业的资金流，盈利能力，供给稳定性等多个方面，这些影响在各行业间也有所差别。我们采用向量自回归（VAR）模型分析突发因素滞后项对未来的影响，得出各行业的脉冲响应图。我们认为附件二中各企业是行业中的典型企业，由此将脉冲响应信号分行业作用在各企业上，计算风险指标并调整贷款策略。

我们的步骤思路如图 6所示：

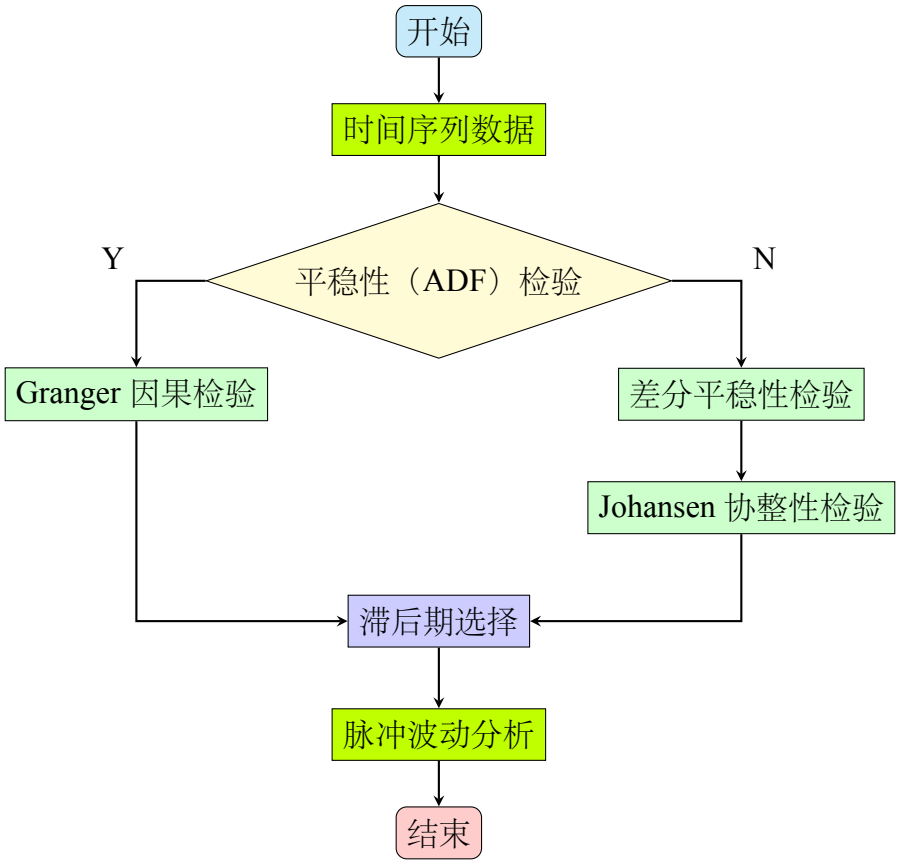

图 **6**　主要建模步骤流程图

VAR 模型可有效刻画相互关联的时间序列系统，进而分析外生冲击对系统的动态扰动。选取各行业股票的价格作为内生变量列，由此可综合覆盖各类突发事件的影响。使用 Choice 金融终端依次选取建筑业指数（399150），电子指数（399136），恒生科技指数（HSTECH）等，个体经营企业则使用中证全指作为时间序列数据（详见支撑材料[数据/股票数据]），通过突发事件内的股价变化反应行业情况的波动。并可根据行业增加值，冲击深度等指标验证模型可行性。

5.4 新冠疫情下的风险评估与信贷调整

5.4.1 VAR 模型建立

平稳性（ADF）检验：

平稳性是传统 VAR 模型乃至多数时间序列模型的基础，因此对数据进行平稳性(ADF) 检验是有必要的。当 t 统计量对应的 p 值小于 0.1 时，可认为时间序列呈现平稳[4]。经检验，原始数据尚不满足平稳性，而经一阶差分后总体满足平稳性，表 6为一阶差分后的 ADF 检验结果：

表 **6**　一阶差分后的 **ADF** 检验结果

行业	t 统计量	p-Value	检验结果
个体	3.010664	0.0147	平稳
医药	3.802687	0.0042	平稳
材料	2.584049	0.0295	平稳
科技	1.633012	0.1369	不平稳
餐饮	4.542944	0.0014	平稳
电子	6.264886	0.0001	平稳
互联网	2.720202	0.0236	平稳
建筑	4.548457	0.0013	平稳
零售商贸	2.092551	0.0659	平稳
运输	2.463188	0.0359	平稳

Johansen 协整检验：

差分方法的缺点是会造成原始信息损失。对于非平稳时间序列，相关学者认为各变量之间存在协整关系即可以直接建立 VAR 模型 [5]。因此常用的要求平稳性的 Granger 因果检验已不适用，本文采用 Johansen 协整检验来验证数据的协整关系 [6]，相关迹统计量和 p 值计算如下：

表 **7**　**Johansen** 协整检验

原假设	特征根	迹统计量	5% 临界值	p 值
None	0.479758	59.406985	51.083569	0.017917
At most 1	0.346109	37.275916	43.516948	0.273684
At most 2	0.219581	32.986310	39.149128	0.239486

5% 的显著性水平下，在“序列间至多存在 1 个协整关系”处首次接受原假设，在“序

列间不存在协整关系处”拒绝了原假设。因此可认为各行业指数间存在协整关系，可建立 VAR 模型。

滞后期的选择:

满足 VAR 模型前提条件后，还需确定滞后期参数以实现具体建立模型。根据对数似然值、似然比、最终预测误差等滞后期选择标准，我们最终确定模型滞后期为 1 阶时效果最好，相关标准计算如下（其中’*’ 上标代表最优阶数位置）:

表 **8** 滞后期的选择

滞后期	对数似然值	似然比	最终预测误差	AIC	SC	HQ
0	1137.72900	NA	2.46E-04	-0.25670	-2.68210	-2.21890
1	2495.28400	4539.59600*	5.36E-06	-11.36940*	-9.29300	-9.85490*
2	2789.31500	821.90000	1.76E-06	-11.35070	-12.82030*	-9.39870
3	3061.58200	241.73000	5.84E-07*	-8.06890	-6.48950	-7.11560

至此已完成 VAR 模型的全部准备，VAR 模型形式如下:

$$\begin{bmatrix} y_{1t} \\ y_{2t} \\ \vdots \\ y_{kt} \end{bmatrix} = A_1 \begin{bmatrix} y_{1t-1} \\ y_{2t-1} \\ \vdots \\ y_{kt-1} \end{bmatrix} + A_2 \begin{bmatrix} y_{1t-2} \\ y_{2t-2} \\ \vdots \\ y_{kt-2} \end{bmatrix} + \cdots + \begin{bmatrix} \varepsilon_{1t} \\ \varepsilon_{2t} \\ \vdots \\ \varepsilon_{kt} \end{bmatrix} \tag{12}$$

这里 y_{it} 即为新冠疫情期间各行业股价的时间序列。

脉冲图分析:

使用乔里斯基正交化残差进行脉冲响应分析，借助 statsmodels 包生成行业的脉冲响应数据图 7。

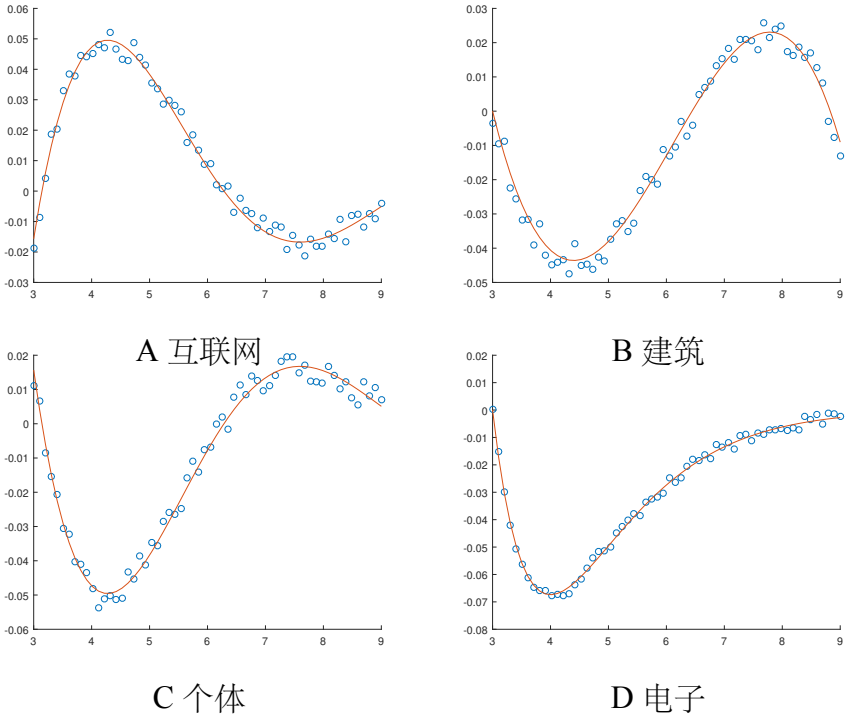

图 **7**　各行业脉冲图

在各企业原有的时间序列预测数据上，导入脉冲响应函数，即可得调整后的企业净值流数据。由此重新进行信用分险评级，计算贷款策略，其计算过程与问题二一致，策略结果如表 9（完整表格见支撑材料中的 [数据/schme3.xlsx]）：

表 **9**　问题三中新冠肺炎冲击下的贷款策略（部分）

企业代号	贷款金额（元）	贷款利率
E124	100000.0	0.08214
E125	727049.1	0.08218
E126	727049.2	0.08453
……	……	……
E423	0.0	0.09855
E424	0.0	0.09855
E425	100000.0	0.06976

5.4.2 行业增加值与冲击深度实证检验

我们从两个角度检验 VAR 模型与实际情况的符合程度。首先，根据国家统计局数据，2020 年 1-2 月相关行业的增加值环比趋势如表 10（全图见支撑材料 [数据/增加值数据]，来源国家统计局）：

表 **10**　行业增加值环比

行业	个体经营	商品零售	···	原材料
增加值	-11.3%	-15.8%	···	-2.7%

另一方面，突发事件对行业领域的冲击，也可依据物理意义上的冲击波模型类比。根据陈菁菁，黄洁等的研究 [7]，其冲击深度指标可表示为：

$$deep = \frac{\Delta V_2}{2\bar{V}_1} = \frac{\max V_2 - \min V_2}{2\bar{V}_1} \tag{13}$$

其中 V_2 选取疫情发生后的股票价格序列，V_1 为疫情发生前的股票价格序列。分行业计算上述指标，疫情爆发时间点选取为 2020 年 1 月 20 日，事件窗口选择为 [-40,100]，用绘制冲击结果深度图如图 8：

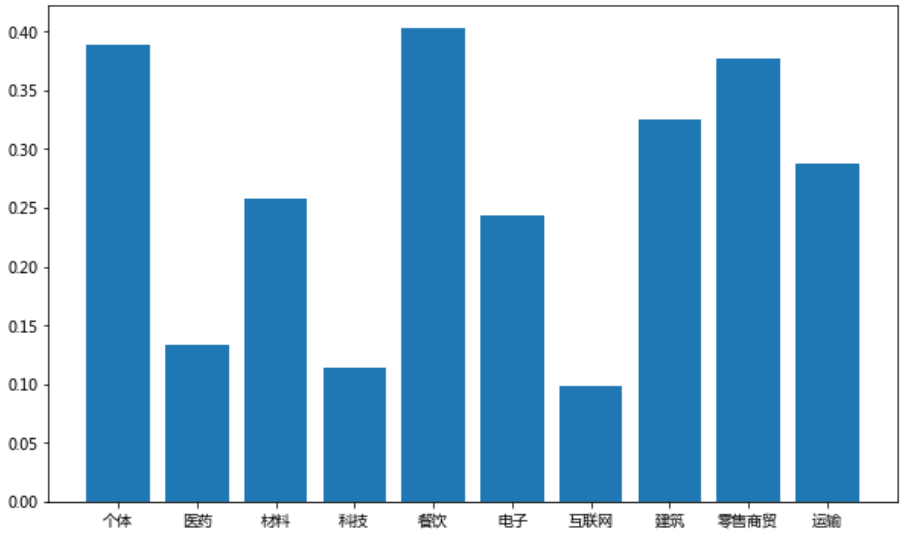

图 **8**　各行业冲击深度

对比上述两指标与 VAR 模型的行业脉冲图，个体，零售等大部分行业受疫情负面冲击，而互联网行业却呈现小幅度的上升趋势，可以认为脉冲响应与行业指标的波动趋势是整体一致的，模型的实际性较为良好。

5.5 次贷危机下的风险评估与信贷调整

次贷危机发生于 2007 至 2009 年。美国房地产泡沫的破裂，席卷了投行，保险等金融机构，进而冲击实体企业部门。我们选取 2007 年 1 月至 2009 年 12 月的股票数据建立 VAR 模型，其建立过程与前文完全一致。从脉冲响应的结果来看，次贷危机几乎对各行业都造成了负面冲击，金融企业负面波动较大，这与新冠疫情的影响有所区别。这也可通过行业增加值和冲击深度指标加以验证。

将脉冲响应信号输入企业的净值流，得出信贷策略如表 11（完整表格见支撑材料中的 [数据/schme4.xlsx]）：

表 **11**　问题三中经济危机冲击下的贷款策略（部分）

企业代号	贷款金额（元）	贷款利率
E124	100000.0	0.07773
E125	745391.1	0.07564
E126	745391.1	0.06601
……	……	……
E423	0.0	0.09892
E424	0.0	0.09892
E425	100000.0	0.07409

六、 模型分析

6.1 模型的优点

1. 从数据挖掘的方向入手，角度新颖，运用时间序列结合资金流量等财务方法对所给数据进行统计分析，可以从海量数据中找到隐含规律;
2. 模型应用较为全面，运用机器学习的模型，既可从企业经营信息拟合预测出企业信用等级，又能根据企业经营状况分析企业体量，从而依据这些信息建立优化模型最大化银行的收益;
3. 模型综合性强，使用方法广泛，不仅用时间序列的分析预测了企业交易数据、交易量等的因素，而且利用随机分析等方法考虑了企业违约和最大收益等关系，并结合物理类比等方法;
4. 模型具有良好的可泛化性，本文所建立的模型和方法可以应用于其他银行和企业的信用等级评估，而不单单局限于题目所给出的企业数据。第三问之中的 VAR 模型基于多维股价时间序列，不仅可以分析新冠疫情和经济危机给企业带来的影响，也可以用于其他多种突发因素的分析和建模;
5. 模型指标具有良好创新性和解释性。使用资金流的月中位值刻画企业资金实力，使用脉冲响应的概念来描述新冠疫情和经济危机给企业和行业带来的影响，简明形象，易于理解。

6.2 模型的缺点

1. 由于附件提供的数据维度较为单一，仅提供了贷款企业的进项发票信息和销项发票信息，而缺少其他的公司财务信息，因此模型的预测准确率收到了一定的影响;
2. 附件提供的公司信息不完整，导致无法准确判断公司所在的行业，只能由公司名称来大致推断企业所在的行业，进而得到企业在疫情中所受到的冲击的大小。

参考文献

[1] Wayne A. Woodward,Henry L. Gray,Alan C Elliott. Applied Time Series Analysis[M].Taylor and Francis:2011-12-02.

[2] Joachims T. Making large-scale SVM learning practical[R]. Technical Report, 1998.

[3] Hastie T, Rosset S, Zhu J, et al. Multi-class adaboost[J]. Statistics and its Interface, 2009, 2(3): 349-360.

[4] 管河山, 邹清明, 罗智超. 时间序列平稳性分类识别研究 [J]. 统计与信息论坛,2016,31(04):3-8.

[5] 钟志威, 雷钦礼.Johansen 和 Juselius 协整检验应注意的几个问题 [J]. 统计与信息论坛,2008(10):80-85+90.

[6] 高铁梅. 计量经济分析方法与建模：EViews 应用及实例：第 2 版［M］. 北京：清华大学出版社，2009：267-318

[7] 陈菁菁, 黄洁. 疫情事件对农产品市场价格冲击的测度 [J]. 统计与决策,2019,35(22):109-112.

[8] 姜启源, 谢金星, 叶俊. 数学模型 [M]. 北京: 高等教育出版社, 2003.

附录

附录清单：

- 附录一：**数据预处理——process.py**
- 附录二：**问题一月序列——t1_sqmonth.py**
- 附录三：**各类企业净值变化——各类企业净值变化.py**
- 附录四：**问题一 ARIMA 预测——t1_arima.py**
- 附录五：**问题一六因子提取——t1_factor.py**
- 附录六：**问题一双模型比较——t1_svmada.py**
- 附录七：**信用等级-违约率——信用等级-违约率.py**
- 附录八：**企业信用等级流失率——企业信用等级流失率.py**
- 附录九：**问题一优化模型求解——t1.m**
- 附录十：**问题一优化目标函数——fun1.m**
- 附录十一：**问题二月序列——t1_sqmonth.py**
- 附录十二：**问题二 ARIMA 预测——t2_arima.py**
- 附录十三：**问题二六因子提取——t2_factor.py**
- 附录十四：**问题二评估等级——t2_predict.py**
- 附录十五：**预测信用等级分布——预测信用等级分布.py**
- 附录十六：**问题二优化模型求解——t2.m**
- 附录十七：**问题二优化目标函数——fun2.m**

以下是本组建模时用到的计算、绘图代码：

- 附录一：**数据预处理——process.py**：

```
def process(filename,sheetname):
    data=pd.read_excel(filename,sheet_name=sheetname)
    data.columns=['企业代号','发票号码','开票日期','单位代号','金额','税额','价税合计','发票状态']
    if data.loc[1,'企业代号']!=1:
        data['企业代号']=data['企业代号']-123#企业代号为了方便，转化为数字
    invalidindex=[]
    n=data['企业代号'].max()+1
    print(n)
    invalid=[0]*n
    for i in range(len(data)):
        if data.loc[i,'发票状态']=='作废发票':
```

```
            invalidindex.append(i)#记录作废发票在记录之中的下标，将相应的记录去掉
    data=data.drop(index=invalidindex)
    data.index=list(range(len(data)))
    group=data.groupby('企业代号')

    minus=[0]*n
    minus_index=[]
    depart_index=[]
    for i in range(len(data)):
        if data.loc[i,'价税合计']<0:
            minus[data.loc[i,'企业代号']]+=data.loc[i,'价税合计']
            minus_index.append(i)# 记录负数发票的下标
    minus_ratio=[0]*n
    for i in range(1,n):
        minus_ratio[i]=abs(minus[i])/group.get_group(i)['价税合计'].sum()# 计算负数发票的比重
    return data,minus_ratio
```

- 附录二: **问题一月序列——t1_sqmonth.py**:

```
#问题预处理
buy,buy_minus_ratio=process('E:\\2020CUMCM\\CUMCM2020Probelms\\C\\data1.xlsx','进项发票信息')
sell,sell_minus_ratio=process('E:\\2020CUMCM\\CUMCM2020Probelms\\C\\data1.xlsx','销项发票信息')

#计算所有企业每一天的的净值序列
# 每一个月取其月中位数作为这一个月净值时间序列数据
buy['tag']=-1*buy['价税合计']#调整符号
sell['tag']=sell['价税合计']
sq_month=[[0]]#每一个元素都是一家企业的月净值信息
for j in range(1,124):
    sq_month.append([0])
    flow=pd.concat([buy[buy['企业代号']==j],sell[sell['企业代号']==j]],axis=0)#整合销方发票和买方发票信息
```

```
    flow=flow.sort_values(by=['开票日期'],ascending=True)#按照日期前后进行排序
    flow['开票日期']=pd.to_datetime(flow['开票日期'])
    begin = flow['开票日期'].min()
    end = flow['开票日期'].max()
    lastmonth=0
    agg=0
    change=[0]
    for i in range((end - begin).days + 1):
        day = begin + datetime.timedelta(days=i)
        if (day.__getattribute__('month')!=lastmonth) and i:#进入一个新的月份
            sq_month[j].append(np.median(change)+agg)
            agg+=change[-1]
            lastmonth=day.__getattribute__('month')
            change=[0]
            change.append(flow.loc[flow['开票日期']==day,'tag'].sum())
        else:
            if day in list(flow['开票日期']):#当日有交易
                change.append(change[-1]+flow.loc[flow['开票日期']==day,'tag'].sum())#记录当日交易给净值带来的变化
            else:
                change.append(change[-1])
    sq_month[j].append(np.median(change)+agg)

#存储月净值序列数据
store=pd.DataFrame(sq_month).T
wb=Workbook()
ws=wb.active
ws.append(list(range(124)))
for i in range(len(store)):
    line=list(store.loc[i])
    ws.append(line)
wb.save('sq_month.xlsx')

```

```
# 记录每一家企业交易记录所在的月份
month=[[0]]
for j in range(1,124):
    month.append([0])
    flow=pd.concat([buy[buy['企业代号']==j],sell[sell['企业代号']==j]],axis=0)#整合销方发票和买方发票信息
    flow=flow.sort_values(by=['开票日期'],ascending=True)#按照日期前后进行排序
    flow['开票日期']=pd.to_datetime(flow['开票日期'])
    begin = flow['开票日期'].min()
    end = flow['开票日期'].max()
    lastmonth=0
    agg=0
    change=[0]
    for i in range((end - begin).days + 1):
        day = begin + datetime.timedelta(days=i)
        if day.__getattribute__('month')!=lastmonth:#进入一个新月份
            month[j].append(str(day.__getattribute__('year'))+'-'+str(day.__getattribute__('month')))#记录新月份
            lastmonth=day.__getattribute__('month')
    month[j].append(str(day.__getattribute__('year'))+'-'+str(day.__getattribute__('month')))

#存储月份信息
store=pd.DataFrame(month).T
wb=Workbook()
ws=wb.active
ws.append(list(range(124)))
for i in range(len(store)):
    line=list(store.loc[i])
    ws.append(line)
wb.save('month.xlsx')
```

- 附录三: 各类企业净值变化——各类企业净值变化.py:

```
import numpy as np
```

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

df = pd.read_excel("~/Desktop/sq_month.xlsx").drop(columns=[0])
df

A_seq = df[2].dropna()
A_seq
plt.plot(A_seq)
plt.show()

B_seq = df[20].dropna()
B_seq
plt.plot(B_seq)
plt.show()

C_seq = df[29].dropna()
C_seq
plt.plot(C_seq)
plt.show()

D_seq = df[99].dropna()
D_seq
plt.plot(D_seq)
plt.show()

l = [A_seq, B_seq, C_seq, D_seq]
title = ['A类企业', 'B类企业', 'C类企业', 'D类企业']
myfont = matplotlib.font_manager.FontProperties(fname = '微软雅黑.ttf'
    , size=15)
matplotlib.rcParams['axes.unicode_minus'] = False

plt.figure(figsize=(16,12))
plt.title("各类企业净值变化",  fontproperties = myfont)
```

```
for i in range(2):
    for j in range(2):
        plt.subplot(2, 2, 2*i+j+1)
        plt.plot(l[2* i + j])
        plt.title(title[2*i+j], fontproperties = myfont)

plt.savefig("各类企业净值变化.png")
plt.show()

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

L = []
for i in range(4):
    _ = []
    for j in range(1, len(l[i])):
        _.append((l[i][j]-l[i][j-1]))
    L.append(pd.DataFrame(_))
L[0]

plot_acf(L[0])
plt.savefig("A类acf.png")
plt.show()

plot_acf(L[1])
plt.savefig("B类acf.png")
plt.show()

plot_acf(L[2])
plt.savefig("C类acf.png")
plt.show()

plot_acf(L[3])
plt.savefig("D类acf.png")
plt.show()
```

- 附录四: 问题一 ARIMA 预测——t1_arima.py:

```
def arima(sample,dx):
    pmax = int(len(dx) / 10)    #阶数超过 length /10,预测效果降低
    qmax = int(len(dx) / 10)
    bic_matrix = []
    for p in range(pmax +1):
        temp= []
        for q in range(qmax+1):
            try:
                temp.append(ARIMA(sample, (p, 1, q)).fit().bic)
            except:
                temp.append(None)
            bic_matrix.append(temp)

    bic_matrix = pd.DataFrame(bic_matrix)
    p,q = bic_matrix.stack().idxmin()  #找出最小值所在位置
    model = ARIMA(sample, (p,1,q)).fit()#模型拟合
    return list(model.forecast(6)[0])

#预测123家企业在未来6个月之中的净值
pred=[[0,0,0,0,0,0]]
for i in range(1,124):
    sample=pd.DataFrame(sq_month.loc[i].dropna())
    dx=sample.diff().dropna()
    sample.columns=['value']
    try:
        one=arima(sample,dx)
    except:
        one=[0,0,0,0,0,0]
    pred.append(one)

#计算六个月净值预测值的中位数和变异系数并进行存储
pred['median']=''
pred['cv']=''
for i in range(123):
```

```
    line=pred.loc[i,['pred1','pred2','pred3','pred4','pred5','pred6']]
    pred.loc[i,'median']=line.median()
    pred.loc[i,'cv']=np.std(line)/line.mean()
    #pred.loc[i,'minus']=sell_minus_ratio[i+1]

from openpyxl import Workbook
wb=Workbook()
ws=wb.active
ws.append(['pred1','pred2','pred3','pred4','pred5','pred6','median','
    cv'])
for i in range(123):
    line=list(pred.loc[i])
    ws.append(line)
wb.save('pred.xlsx')
```

- 附录五: 问题一六因子提取——t1_factor.py:

```
#计算峰度和偏度并存储
kurtlist=[0]
for i in range(1,124):
    part_sell=sell.loc[sell['企业代号']==i]
    group=part_sell.groupby('单位代号')##按照买方单位代号进行分组
    kurtlist.append(group['价税合计'].sum().kurt())

skewlist=[0]
for i in range(1,124):
    part_buy=buy.loc[buy['企业代号']==i]
    skewlist.append(part_buy['价税合计'].skew())

wb=Workbook()
ws=wb.active
ws.append(['skew','kurt'])
for i in range(1,124):
    line=[skewlist[i],kurtlist[i]]
    ws.append(line)
wb.save('峰度偏度.xlsx')
```

```
20
21 #计算负数发票占比，月亏空，供求关系的峰度和偏度
22 pred=pd.read_excel('pred.xlsx')
23 month_loss=(pd.read_excel('month_loss.xlsx')).T
24 svmdata=pred.drop(columns=['pred1','pred2','pred3','pred4','pred5','
    pred6'])
25 svmdata['minus']=''
26 svmdata['loss']=''
27 svmdata['skew']=''
28 svmdata['kurt']=''
29 #svmdata['flow']=''
30 svmdata['rank']=''
31 for i in range(123):
32     svmdata.loc[i,'minus']=sell_minus_ratio[i+1]
33     svmdata.loc[i,'loss']=np.max(month_loss.loc[i+1])
34     #svmdata.loc[i,'flow']=sell.loc[sell['企业代号']==i+1,'价税合计'].
    sum()+buy.loc[buy['企业代号']==i+1,'价税合计'].sum()
35 sk=pd.read_excel('峰度偏度.xlsx')
36 svmdata['skew']=sk['skew']
37 svmdata['kurt']=sk['kurt']
38 svmdata['rank']=sk['评级']
39
40 #将计算结果进行存储
41 from openpyxl import Workbook
42 wb=Workbook()
43 ws=wb.active
44 ws.append(['median','cv','minus','loss','skew','kurt','rank'])
45 for i in range(len(svmdata)):
46     ws.append(list(svmdata.loc[i]))
47 wb.save('factor1.xlsx')
```

- 附录六: **问题一双模型比较——t1_svmada.py**:

```
1 import sklearn
2 from sklearn import svm
3 from sklearn import model_selection
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import recall_score,precision_score,
    accuracy_score
# 进行数据预处理
x,y=np.split(svmdata,indices_or_sections=(6,),axis=1) #x为数据，y为标
    签
x['skew']=x['skew'].fillna(0)
x['kurt']=x['kurt'].fillna(0)#峰度和偏度的缺失值全部填充为0
x['cv']=np.abs(x['cv'])
x['cv']=(x['cv']-np.min(x['cv']))/(np.max(x['cv'])-np.min(x['cv']))
#x['flow']=(x['flow']-np.min(x['flow']))/(np.max(x['flow'])-np.min(x['
    flow']))
#x=x.drop(columns=['cv'])
x['median']=(x['median']-np.min(x['median']))/(np.max(x['median'])-np.
    min(x['median']))#数据归一化操作
#x=x.drop(columns=['median'])
x['loss']=(x['loss']-np.min(x['loss']))/(np.max(x['loss'])-np.min(x['
    loss']))
x=x.drop(columns=['loss'])
x['minus']=(x['minus']-np.min(x['minus']))/(np.max(x['minus'])-np.min(
    x['minus']))
#x=x.drop(columns=['minus'])
x['skew']=(x['skew']-np.min(x['skew']))/(np.max(x['skew'])-np.min(x['
    skew']))
#x=x.drop(columns=['skew'])
x['kurt']=(x['kurt']-np.min(x['kurt']))/(np.max(x['kurt'])-np.min(x['
    kurt']))
#分割训练集和测试集
train_data,test_data,train_label,test_label =sklearn.model_selection.
    train_test_split(x,y, random_state=1, train_size=0.7,test_size=0.3)


#尝试使用SVM进行拟合
classifier=svm.SVC(C=2,kernel='rbf')
classifier.fit(train_data,train_label) #SVM拟合
```

```
pred_label=classifier.predict(test_data)#SVM预测
print(recall_score(test_label,pred_label,average='micro'))
print(precision_score(test_label,pred_label,average='micro'))

#Adaboost尝试进行拟合
from sklearn.ensemble import AdaBoostClassifier
from sklearn import metrics
bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=10,
    min_samples_split=5, min_samples_leaf=5),
                         algorithm="SAMME",
                         n_estimators=250, learning_rate=0.20)#设置
    Adaboost分类器
bdt.fit(train_data, train_label)
pred_label=bdt.predict(test_data)
print(accuracy_score(test_label,pred_label))
print(recall_score(test_label,pred_label,average='micro'))
print(precision_score(test_label,pred_label,average='micro'))
print(metrics.confusion_matrix(test_label, pred_label, sample_weight=
    None))#打印混淆矩阵

# 使用PCA对几个因子进行主成因子分析
from sklearn.decomposition import PCA
pca = PCA(n_components=4)
trim= pca.fit(x).transform(x)
print(x.shape)#原始变量维度
print(trim.shape)#主成分变量的维度
trim=pd.DataFrame(trim)
print('各主成分贡献度:{}'.format(pca.explained_variance_ratio_))
train_data,test_data,train_label,test_label =sklearn.model_selection.
    train_test_split(trim,y,random_state=1, train_size=0.7,test_size
    =0.3)

#进行了PCA之后重新进行Adaboost拟合
from sklearn.ensemble import AdaBoostClassifier
bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=11,
```

```
    min_samples_split=5, min_samples_leaf=10),
                            algorithm="SAMME",
                            n_estimators=150, learning_rate=0.25)
bdt.fit(train_data, train_label)
pred_label=bdt.predict(test_data)
print(accuracy_score(test_label,pred_label))
print(recall_score(test_label,pred_label,average='micro'))
print(precision_score(test_label,pred_label,average='micro'))
final_clf=bdt
```

- 附录七: 信用等级-违约率——信用等级-违约率.py:

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

df = pd.read_excel("~/Desktop/append1.xlsx")
df

A = df.loc[df["信誉评级"]=="A"].copy()
B = df.loc[df["信誉评级"]=="B"].copy()
C = df.loc[df["信誉评级"]=="C"].copy()
D = df.loc[df["信誉评级"]=="D"].copy()

yA = list(A.是否违约).count('是')
nA = list(A.是否违约).count('否')
A_percent = yA/(yA + nA)
A_percent

yB = list(B.是否违约).count('是')
nB = list(B.是否违约).count('否')
B_percent = yB/(yB + nB)
B_percent

yC = list(C.是否违约).count('是')
nC = list(C.是否违约).count('否')
```

```
C_percent = yC/(yC + nC)
C_percent

yD = list(D.是否违约).count('是')
nD = list(D.是否违约).count('否')
D_percent = yD/(yD + nD)
D_percent

x = ['A','B','C','D']
y = [A_percent, B_percent, C_percent, D_percent]

import matplotlib
# 设置图形的显示风格
myfont = matplotlib.font_manager.FontProperties(fname = '微软雅黑.ttf'
    , size=15)
matplotlib.rcParams['axes.unicode_minus'] = False

plt.figure(figsize=(9,6))

plt.bar(x, y, facecolor = '#66AABB', edgecolor = 'black')
plt.title('各信用等级违约率', fontproperties = myfont)

labelfont = {'family' : 'Microsoft YaHei',
'weight' : 'normal',
'size'   : 15,
}

plt.xlabel("信用等级", labelfont)
plt.ylabel("违约率", labelfont)
plt.savefig("各信用等级违约率.png",dpi=300)
plt.show()
```

- 附录八: 企业信用等级流失率——企业信用等级流失率.py:

```
import numpy as np
import pandas as pd
```

```
import matplotlib
import matplotlib.pyplot as plt

df = pd.read_excel("append3.xlsx",header = 1)
df.columns=["年利率", "A", "B", "C"]
df

myfont = matplotlib.font_manager.FontProperties(fname = '微软雅黑.ttf'
    , size=15)
matplotlib.rcParams['axes.unicode_minus'] = False

plt.figure(figsize=(18,6))

title=['A类企业', 'B类企业', 'C类企业']
x = [0.001*i+0.04 for i in range(110)]
yA = [-76.41*_*_+21.98*_-0.6971 for _ in x]
yB = [-67.93*_*_+20.21*_-0.6504 for _ in x]
yC = [-63.94*_*_+19.57*_-0.6393 for _ in x]

Y=[yA, yB, yC]

for i in range(3):
    plt.subplot(1,3, i+1)
    plt.scatter(df["年利率"], df.iloc[:,i+1])
    plt.plot(x, Y[i])
    plt.title(title[i], fontproperties = myfont)

plt.savefig("企业信用等级流失率.png")
```

- 附录九: **问题一优化模型求解——t1.m**:

```
global data
data=xlsread('E:\\2020CUMCM\\CUMCM2020Probelms\\C\\data1.xlsx');
data=[data(:,3)]%加载附件一的信用评级数据，并设为全局变量

lb=[zeros(1,123) ones(1,123)*0.04];
```

```
pred=xlsread('pred.xlsx')
ub=zeros(1,123);
for i =1:123
    ub(i)=max(pred(i,:));
end%依据未来六个月的精致预测来确定贷款额度的上界
ub=[ub 0.15*ones(1,123)];
A=[ones(1,123) zeros(1,123)];
b=100000000% b=M
X0=[ones(1,123)*800000 ones(1,123)*0.1];

for i=1:246
    if ub(i)<lb(i)
        ub(i)=lb(i);
    end%保证上界大于下界
    if i<=123 && data(i)==4
        ub(i)=0;
        lb(i)=0;
    end
    if i<123 && ub(i)>1000000
        ub(i)=1000000;
    end%额度不能超过1000000
end

options=optimoptions(@fmincon,'MaxFunEvals',500000,'MaxIter',10000);
[x,y]=fmincon('fun',X0,A,b,[],[],lb,ub,'',options);
money=x(1:123);
rate=x(124:246);
schme=[money',rate'];
```

- 附录十: 问题一优化目标函数——**fun1.m**:

```
function  f  = fun(x)
sum=0;
global data
for i=1:123
    money=x(i);
```

```
    rate=x(i+123);
    if data(i)==1
        sum=sum+money*rate*(76.41*rate*rate-21.98*rate+1.6971);
    end
    if data(i)==2
        sum=sum+money*(67.93*rate*rate-20.21*rate+1.6504)
   *(-0.0263+0.9737*rate);
    end
    if data(i)==3
        sum=sum+money*(63.94*rate*rate-19.57*rate+1.6393)
   *(-0.0588+0.9412*rate);
    end


end
f=-sum;
```

- 附录十一: **问题二月序列——t1_sqmonth.py**:

```
#导入数据并进行预处理
buy,buy_minus_ratio=process('E:\\2020CUMCM\\CUMCM2020Probelms\\C\\
   data2.xlsx','进项发票信息')
sell,sell_minus_ratio=process('E:\\2020CUMCM\\CUMCM2020Probelms\\C\\
   data2.xlsx','销项发票信息')

#计算302家企业每一个月的净值并存储
import datetime
buy['tag']=-1*buy['价税合计']
sell['tag']=sell['价税合计']
sq_month=[[0]]
for j in range(1,303):
    sq_month.append([0])
    flow=pd.concat([buy[buy['企业代号']==j],sell[sell['企业代号']==j
   ]],axis=0)# 将销方发票和买方发票进行整合
    flow=flow.sort_values(by=['开票日期'],ascending=True)#按照日期进行
   排序
```

```
    flow['开票日期']=pd.to_datetime(flow['开票日期'])
    begin = flow['开票日期'].min()
    end = flow['开票日期'].max()
    lastmonth=0
    agg=0
    change=[0]
    for i in range((end - begin).days + 1):
        day = begin + datetime.timedelta(days=i)
        if (day.__getattribute__('month')!=lastmonth) and i:#进入一个新的月份
            sq_month[j].append(np.median(change)+agg)
            agg+=change[-1]
            lastmonth=day.__getattribute__('month')
            change=[0]
            change.append(flow.loc[flow['开票日期']==day,'tag'].sum())
        else:
            if day in list(flow['开票日期']):#当日有交易记录
                change.append(change[-1]+flow.loc[flow['开票日期']==day,'tag'].sum())
            else:
                change.append(change[-1])
    sq_month[j].append(np.median(change)+agg)


store=pd.DataFrame(sq_month).T
from openpyxl import Workbook
wb=Workbook()
ws=wb.active
ws.append(list(range(0,303)))
for i in range(len(store)):
    ws.append(list(store.loc[i]))
wb.save('sq_month2.xlsx')
```

- 附录十二: **问题二 ARIMA 预测——t2_arima.py**:

```
#该函数接受时间序列数据及其一阶差分作为数据，训练得到ARIMA模型
```

```
def arima(sample,dx):
    pmax = int(len(dx) / 10)    #一般阶数不超过 length /10
    qmax = int(len(dx) / 10)
    bic_matrix = []
    for p in range(pmax +1):
        temp= []
        for q in range(qmax+1):
            try:
                temp.append(ARIMA(sample, (p, 1, q)).fit().bic)
            except:
                temp.append(None)
            bic_matrix.append(temp)

    bic_matrix = pd.DataFrame(bic_matrix)
    p,q = bic_matrix.stack().idxmin()  #找出最小值所在位置
    model = ARIMA(sample, (p,1,q)).fit()#模型拟合
    return list(model.forecast(6)[0])



#使用ARIMA预测未来六个月的公司净值
from statsmodels.tsa.arima_model import ARIMA
pred=[[0,0,0,0,0,0]]
for i in range(1,303):
    sample=pd.DataFrame(sq_month[i]).dropna()
    dx=sample.diff().dropna()
    sample.columns=['value']
    try:
        one=arima(sample,dx)
    except:
        one=[0,0,0,0,0,0]
    pred.append(one)

from openpyxl import Workbook
wb=Workbook()
```

```
ws=wb.active
ws.append(['pred1','pred2','pred3','pred4','pred5','pred6'])
for i in range(len(pred)):
    ws.append(list(pred[i]))
wb.save('pred2.xlsx')
```

- 附录十三: **问题二六因子提取——t2_factor.py**:

```
# 计算信用评级影响因子
pred=pd.read_excel('pred2.xlsx')
factor=pred
factor['median']=''
factor['cv']=''
factor['minus']=''
factor['skew']=''
factor['kurt']=''
for i in range(1,303):
    line=pred.loc[i,['pred1','pred2','pred3','pred4','pred5','pred6']]
    #line.columns=['value']
    factor.loc[i,'median']=line.median()
    factor.loc[i,'cv']=np.std(line)/line.mean()
    factor.loc[i,'skew']=buy.loc[buy['企业代号']==i,'价税合计'].skew()
    part_sell=sell.loc[sell['企业代号']==i]
    group=part_sell.groupby('单位代号')
    factor.loc[i,'kurt']=group['价税合计'].sum().kurt()
    factor.loc[i,'minus']=sell_minus_ratio[i]
factor['skew']=factor['skew'].fillna(0)
factor['kurt']=factor['kurt'].fillna(0)#缺失值填充
factor=factor.drop(index=[0])#第一列是为了补齐下标而填充的，应当去掉
factor.index=list(range(len(factor)))
factor=factor.drop(columns=['pred1','pred2','pred3','pred4','pred5','
    pred6'])
```

- 附录十四: **问题二评估等级——t2_predict.py**:

```
#进行数据归一化处理
factor['median']=(factor['median']-np.min(factor['median']))/(np.max(
```

```
    factor['median'])-np.min(factor['median']))
factor['cv']=(factor['cv']-np.min(factor['cv']))/(np.max(factor['cv'])
    -np.min(factor['cv']))
factor['skew']=(factor['skew']-np.min(factor['skew']))/(np.max(factor[
    'skew'])-np.min(factor['skew']))
factor['kurt']=(factor['kurt']-np.min(factor['kurt']))/(np.max(factor[
    'kurt'])-np.min(factor['kurt']))#归一化
factor['minus']=(factor['minus']-np.min(factor['minus']))/(np.max(
    factor['minus'])-np.min(factor['minus']))

#使用第一问得到的最好的分类器进行预测，并将结果保留
final_clf.fit(x,y)
#factor=factor.drop(columns=['minus'])
rating=final_clf.predict(factor.iloc[:,[0,1,2,3,4]])
factor['rank']=''
factor['rank']=pd.DataFrame(rating)

from openpyxl import Workbook
wb=Workbook()
ws=wb.active
ws.append(['median','cv','skew','kurt','minus','rank'])
for i in range(len(factor)):
    ws.append(list(factor.loc[i]))
wb.save('factor2.xlsx')
```

- 附录十五: 预测信用等级分布——预测信用等级分布.py:

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

df = pd.read_excel("~/Desktop/factor2.xlsx")
df

# rank分布饼图
```

```
10
11 myfont = matplotlib.font_manager.FontProperties(fname = '微软雅黑.ttf'
       , size=15)
12
13 # rank统计
14 A = list(df['rank']).count('A')
15 B = list(df['rank']).count('B')
16 C = list(df['rank']).count('C')
17 D = list(df['rank']).count('D')
18
19 total = df['rank'].count()
20
21 # 标签
22 labels = ['A', 'B', 'C','D']
23 sizes = [A, B, C, D]
24 colors = ['cornflowerblue','pink', 'blue', 'red']
25 explode = (0, 0.1, 0, 0.1)
26
27 fig1, ax1 = plt.subplots(figsize = (6, 6))
28 # 饼图
29 patches, l_text, p_text = ax1.pie(sizes, explode = explode,
30                                   labels = labels, colors = colors,
31                                   autopct = '%1.1f%%', shadow =
       False,
32                                   startangle = 90)
33 for t in l_text:
34         t.set_size(15)
35 for t in p_text:
36         t.set_size(15)
37 # 文字
38 ax1.axis('equal')
39 plt.title('预测企业信用等级分布图', fontproperties = myfont)
40 plt.legend(loc='best', fontsize = 12)
41 plt.savefig("预测企业信用等级分布图.png", dpi = 300)
42 plt.show()
```

- 附录十六: **问题二优化模型求解——t2.m**:

```
global data
data=xlsread('E:\\2020CUMCM\\CUMCM2020Probelms\\C\\data2.xlsx');
%加载附件二的信用评级预测数据，并设为全局变量

lb=[zeros(1,302) ones(1,302)*0.04];
pred=xlsread('pred2.xlsx');
pred=pred(2:303,:);
ub=zeros(1,302);
for i =1:302
    ub(i)=max(pred(i,:));
end%加载净值预测值作为贷款额度的上界
ub=[ub 0.15*ones(1,302)];
A=[ones(1,302) zeros(1,302)];
b=100000000% b=M
X0=[ones(1,302)*800000 ones(1,302)*0.1];
%设置初始值

for i=1:604
    if ub(i)<lb(i)
        ub(i)=lb(i);
    end
    if i<=302 && data(i)==4
        ub(i)=0;
        lb(i)=0;
    end%对于D等级不发放贷款，上下界都是0
    if i<302 && ub(i)>1000000
        ub(i)=1000000;
    end%额度上界不能超过1000000
end

options=optimoptions(@fmincon,'MaxFunEvals',5000,'MaxIter',10000);
[x,y]=fmincon('fun2',X0,A,b,[],[],lb,ub,'',options);
money=x(1:302);
rate=x(303:604);
```

```
35 schme=[money',rate'];
```

- 附录十七: 问题二优化目标函数——**fun2.m**:

```
1  function  f  = fun2(x)
2  sum=0;
3  global data
4  for i=1:302
5      money=x(i);
6      rate=x(i+302);
7      if data(i)==1
8          sum=sum+money*rate*(76.41*rate*rate-21.98*rate+1.6971);
9      end
10     if data(i)==2
11         sum=sum+money*(67.93*rate*rate-20.21*rate+1.6504)
       *(-0.0263+0.9737*rate);
12     end
13     if data(i)==3
14         sum=sum+money*(63.94*rate*rate-19.57*rate+1.6393)
       *(-0.0588+0.9412*rate);
15     end
16
17 end
18 f=-sum;
```