基于数据挖掘的银行信贷策略模型

# 摘要

在市场经济的体制中，银行对中小微型企业的信贷政策对市场经济起推动作用。银行对中小型企业的信贷政策有重大意义：获取合理的融资有助于中小型企业的成长；给予中小企业融资有助于参与中小企业发展，资金提供方能分享发展收益；将社会资源合理配置到生产效率和创新能力较高的中小企业，有助于国家经济的整体发展。

在中小微型企业的信贷决策问题中，本文主要研究了三方面问题：基于企业信誉和评价企业经济实力的银行信贷策略（是否放贷及贷款额度、利率），在年度信贷总额有限情况下的银行信贷调整策略，在突发事件影响下的信贷调整策略。在三方面问题中，我们分别基于风险调整资本回报率RAROC建立风险收益评估模型，基于神经网络的信誉和违约预测模型，基于词向量分类-加权影响模型。

针对问题一，基于风险调整资本回报率（RAROC）公式，推出适用于本题的风险调整资本回报率模型，应用概率论和数理统计相关知识，分析附件1中123家企业进项和销项发票信息，计算出经风险调整的收益和贷款占用的经济成本。其中计算收益时，为预测企业还款能力，利用长短期记忆人工神经网络LSTM建立时间序列模型。对于风险调整资本回报函数，利用爬山法对其求达到最大值时的贷款额度和利率，作为银行对这些企业的信贷策略。最后可视化银行对各个企业放贷策略的结果。

针对问题二，为了补充无信贷记录的企业样本数据中信誉评价的违约记录的缺失特征，建立基于神经网络的信誉预测模型，输入层为每一天的企业现有利润，输出层为信誉等级A/B/C/D的权值和违约概率，训练神经网络来预测附件2中的企业信誉评级。利用填补缺失值后的数据，输入问题一的模型计算出302家企业理论贷款额度。结果数据显示，所需贷款总额超出1亿元，需要进行额度削减。通过基于历史增长数据评价企业资金增长潜力的减额策略，为发展稳定、更具潜力的企业提供更高额度的贷款和相较于其他企业较低的利率。最后可视化贷款额度削减结果。

针对问题三，人为将企业分为个体经营，商业贸易，服务业，工程建设，医药，高新技术，文体和日常生活八类，再通过训练wiki语料数据库建立Word2Vec模型，构造词袋并设定聚类中心，依据词向量欧式距离为302家企业进行分类，得出每个企业作为每一行业类型的概率向量。依据资料的4大突发事件，分别建立影响矩阵、评估特定事件程度，构造影响函数模型，支持评价对特定企业在特定突发事件中的影响系数。最后给出面对突发事件时，银行放贷的调整策略。此外，以2020新冠疫情数据为例验证量化模型的准确性，并给出贷款额度削减结果。

关键词：风险调整资本回报率；LSTM；爬山法；神经网络；词袋模型；Word2Vec

# 一．问题重述

## 1.1 问题背景

信贷市场是信贷工具的交易市场而信贷市场是商品经济发展的产物。在商品经济条件下，随着商品流通的发展，生产日益扩大和社会化，社会资本的迅速转移，多种融资形式和信用工具的运用和流通，促进信贷市场的形成，而商品经济持续、稳定协调发展，又离不开完备的信贷市场体系的支持。

在实际中，由于中小微企业规模相对较小，也缺少抵押资产，因此银行通常是依据信贷政策、企业的交易票据信息和上下游企业的影响力，向实力强、供求关系稳定的企业提供贷款，并可以对信誉高、信贷风险小的企业给予利率优惠。银行首先根据中小微企业的实力、信誉对其信贷风险做出评估，然后依据信贷风险等因素来确定是否放贷及贷款额度、利率和期限等信贷策略。

## 1.2 问题重述

问题一考察对企业信贷风险的量化分析，处理附件1中123家企业信息以及进项销项的发票，给出该银行在年度信贷总额固定时对这些企业的信贷策略，包括是否放贷及贷款额度、利率和期限等。

问题二需要在问题一的基础上，对附件2中302家企业的信贷风险进行量化分析，首先通过分析问题一的结果，推算出银行对中小微企业的信贷评级标准，根据这套标准对附件2中的企业进行评级，并给出该银行在年度信贷总额为1亿元时对这些企业的信贷策略。

问题三提出了突发因素对企业的生产经营和经济效益可能会产生一定的影响，突发因素往往对不同行业、不同类别的企业会有不同的影响。我们首先需要对突发因素和企业进行分类，综合考虑附件2中各企业的信贷风险和可能的突发因素（例如：新冠病毒疫情）对各企业的影响，给出该银行在年度信贷总额为1亿元时的信贷调整策略。

# 二．模型假设

1. 假设每一家企业的每一张发票信息都真实可信。
2. 假设通过机器学习预测企业利润走势图准确。
3. 假设还款是企业资金的期望满足正态分布。
4. 假设企业在经营过程中无倒闭，转让现象发生。
5. 假设没有其他银行参与信贷活动。
6. 假设wiki语料库足够大足以训练出较为可信的模型。

# 三．符号说明及名词定义

|  |  |
| --- | --- |
| 变量 | 说明 |
|  | 经风险调整的收益 |
|  | 银行对企业的贷款额度 |
|  | 企业预期还款额度 |
|  | 信誉系数 |
|  | 贷款年利率 |
|  | 银行预期损失额度 |
|  | 客户流失率 |
|  | 企业还款概率 |
|  | 贷款占用的经济成本 |
|  | 企业利润和供求关系稳定率 |
|  | 客户不流失的期望利润 |
|  | 客户流失的期望利润 |
|  | 突发事件对企业影响函数 |
|  | 资金增长率 |

# 四．模型的建立与求解

## 4.1 问题一的分析与建模

### 4.1.1 问题分析

该问题要求给出该银行在年度信贷总额固定时对123家企业的信贷策略。首先分析处理附件1中企业进项和销项发票信息，基于风险调整资本回报率RAROC建立风险收益评估模型，来衡量承担风险下赚取回报的期望，通过爬山法求得银行期望收益达到最大时，为各个企业提供的贷款额度，利率和期望收益，可视化结果数据。

### 4.1.2 风险调整资本回报率模型建立

风险调整资本回报率为：

在中小微企业的信贷决策问题中，经风险调整的收益包括两个部分，分别为银行理论上应得到的净收益和银行预期损失的收益。

根据概念，经风险调整的收益为：

其中，为银行企业贷款额度，为利率。

银行放贷的收益与客户是否借贷有关，假设客户流失率为，与年贷款利率和客户的信誉评级（credit-rating）有关。为客户不流失的期望利润；为客户流失的期望利润，显然为。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 期望利润 |  |  |

表 1概率分布

计算，为企业无法偿还贷款金额的概率，即企业在还款时限时资金不足的概率（图3解释求法）；当企业有能力偿还时，银行收益为利息，即；否则损失：

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 期望收益 |  |  |

表 2 概率分布

则

整理得：

其中银行预计损失额度为本息与企业无法偿还时抵债资金的差：

其中，在企业无法还款的情况下，企业的预期还款金额为；为信誉系数。

由信誉评价等级和违约记录综合评价而成。

则矩阵为

其中为信誉评级，为是否违约。

即为时的期望值，其中为还款期限时的企业资金（企业的还贷能力与企业的利润是成正相关的），建立的表达式：

其中为还款期限时企业资金的分布函数和概率密度函数。

建立时间序列预测模型，基于3年数据预测未来（还款期限）时企业资金；模型基于长短期记忆人工神经网络（LSTM），是一种时间循环神经网络，通过forget gate能解决循环神经网络（RNN）长期依赖问题。

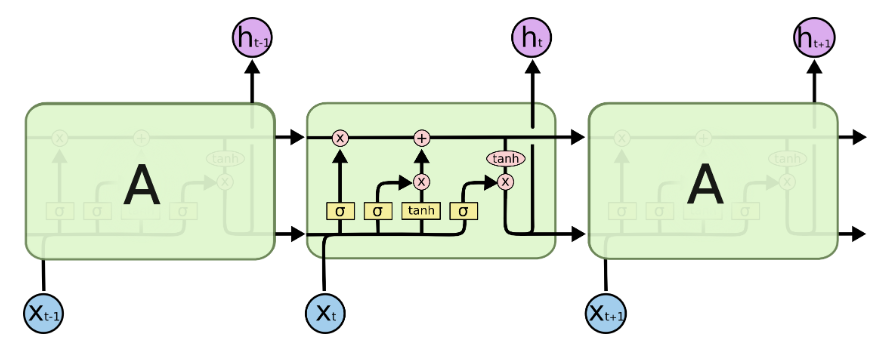


图 1 LSTM模型图示

对附件1中各个企业的进账和销账的发票信息进行数据处理，获取2017年-2020年每日进项价税合计与销项金额。

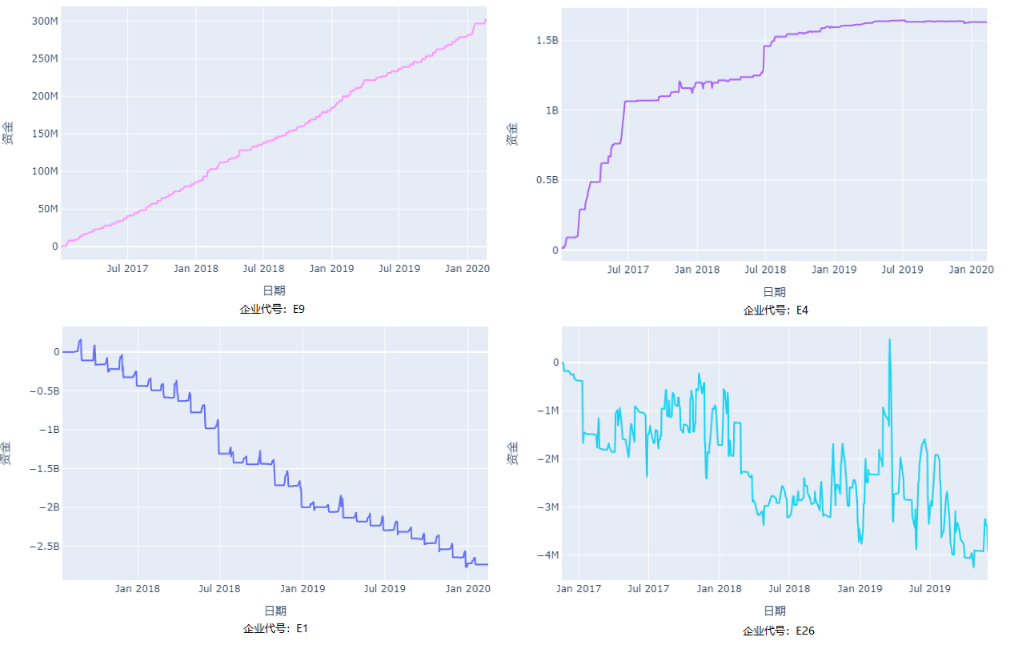


图 2部分企业利润数据走势图

先以2017-2018为训练集，2019为测试集，得模型准确率为；再使用所有数据，以时间为特征、利润为标签的为训练集，预测未来1年的利润（蓝线表示原有数据，红线表示预测值）：

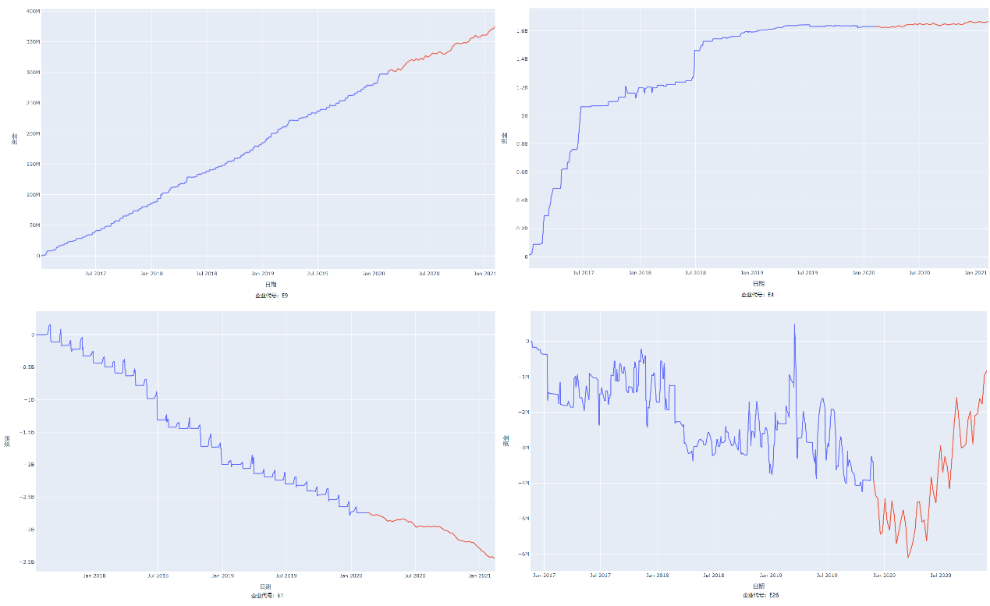


图 3部分企业利润预测图

由LSTM模型在2021年1月（还款期限）预测值和概率得，即还款期限时企业资金期望值和方差，令

基于正态分布构造：

即对于函数上任意一点，还款时企业资金为的概率为，模型图为：

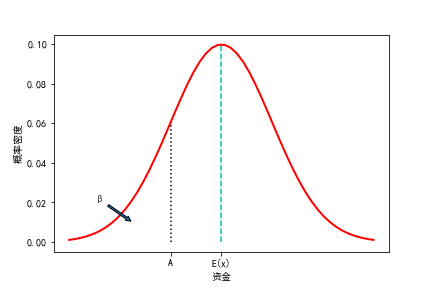


图 4利润概率密度模型与期望资金、贷款额度、无法偿还率的关系

估计无法偿还率需要依据企业利润和供求关系稳定率，企业的大买家（利润贡献率超过5%）利润贡献越大，供求关系越稳定，越大，定义为

即当利润贡献率超过5%的大买家贡献率总和超过30%，表示供求关系很稳定；反之则较不稳定。下图为若干企业的供求关系计算图：

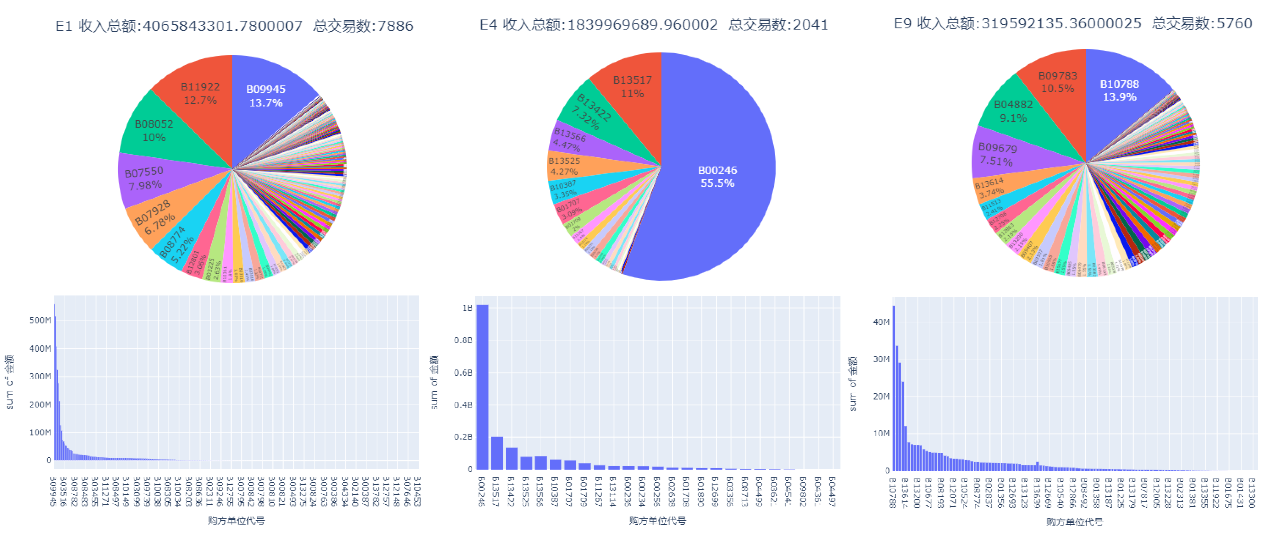


图 5部分企业供求关系统计图

定义为

将表达式进一步化简得：

根据经济学概念，贷款占用的经济成本为：

其中银行预期损失额度的方差为：

整理以上式子可得改进的适用于本题的风险调整资本回报率（RAROC）为：

综上所述，RAROC是关于的二元函数，其中为连续性，为离散型，与参数关系如下：



图 6 函数变量构成和关系

### 4.1.3 爬山法求解最大化RAROC的放贷策略

本题站在银行得角度，希望能获得最大的收益，RAROC正是反应收益的关于的函数。在我们的模型中，银行不为1年后期望企业资金为负或信誉评价为D的企业提供放贷。因此我们需要找到对于每个企业的参数，使得银行从该企业获得的期望收益最大。



图 7求解极大时取值算法

其中的可能值在附件3中的数据已经体现，通过枚举可以使RAROC函数转换为关于的一元函数，使用爬山法（Climbing）对此一元函数求最大值。

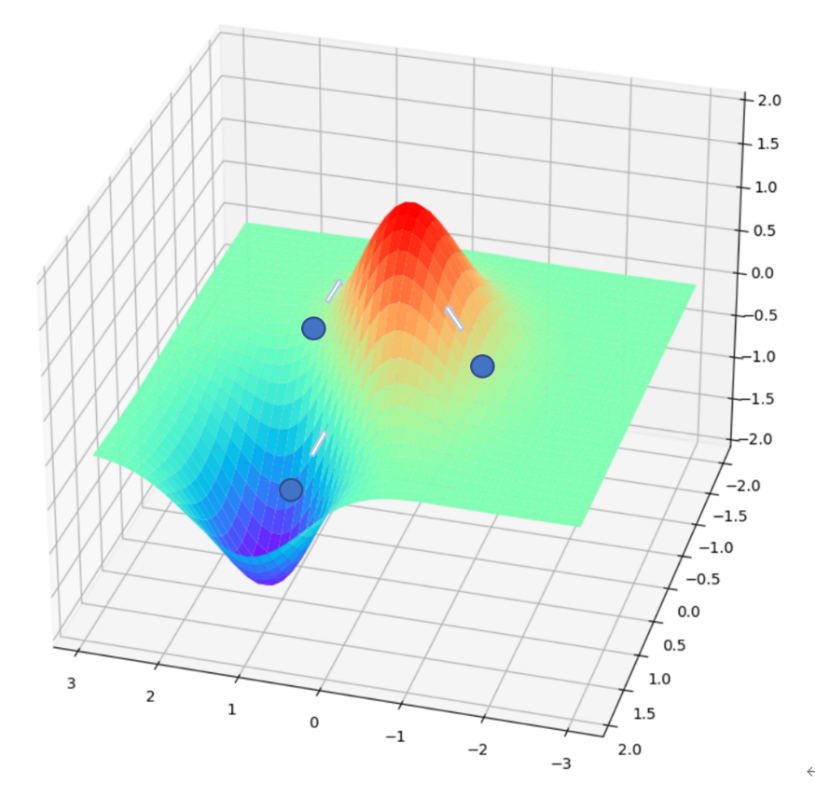


图 8爬山法示意图

爬山法假设当前解和周围的解是有变化规律的，当前解得下方有一个函数值较大的解，则认为沿着这个方向走，解会越来越大。具体步骤为：选择若干解作为种子解，每次寻找与这个解相近的解，如果相近的解中有函数值更大的解，则把这个解作为种子解。而如果周围的解都比该解的代价大，则表示已经到达了局部极大值点，搜索停止。比较若干个种子到达的解，得出的最优值。

求得各企业使最大的，可视化展示所得结果。其中横坐标的贷款额度，纵坐标是年利率，点的面积是（风险期望收益）的值。

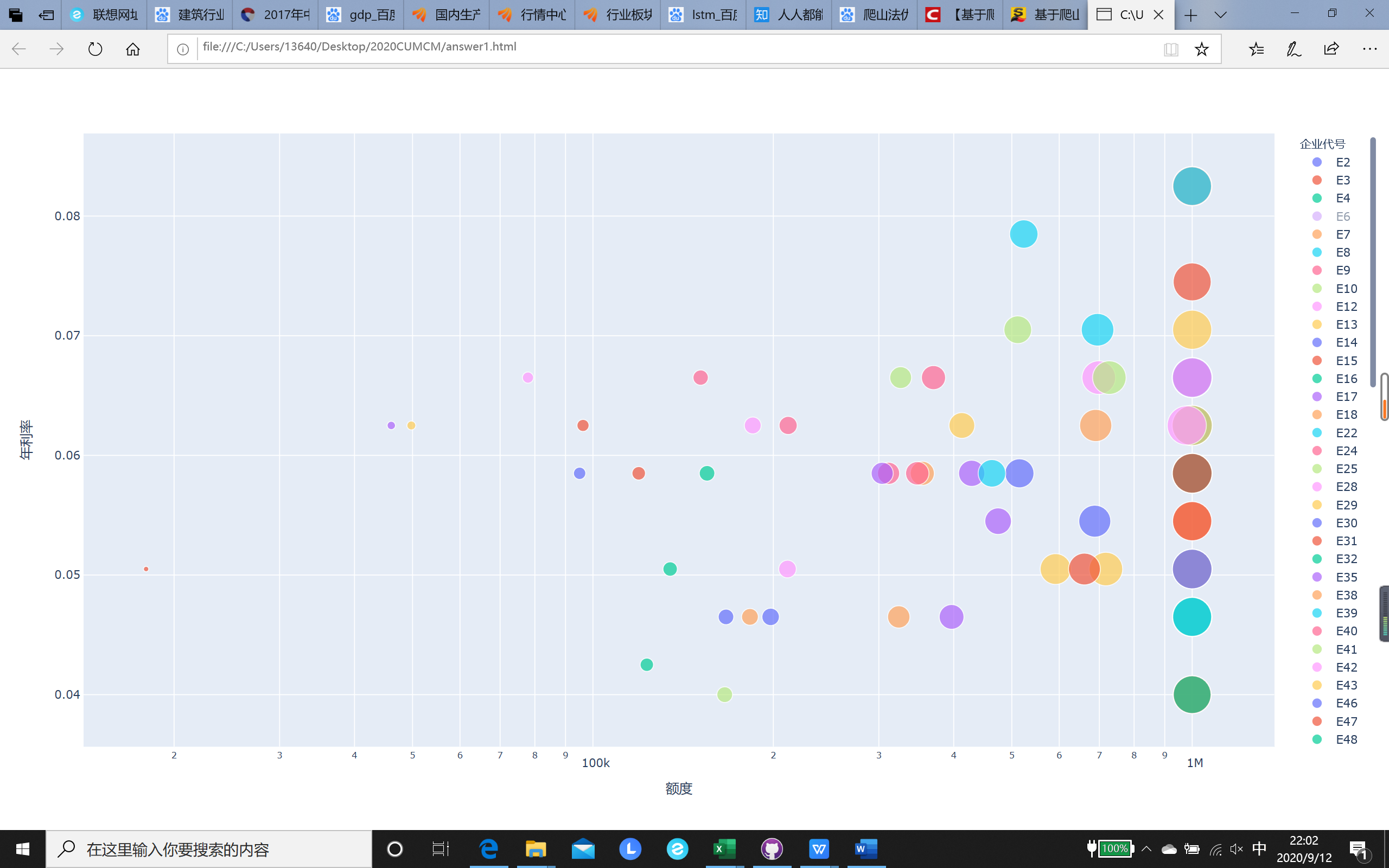
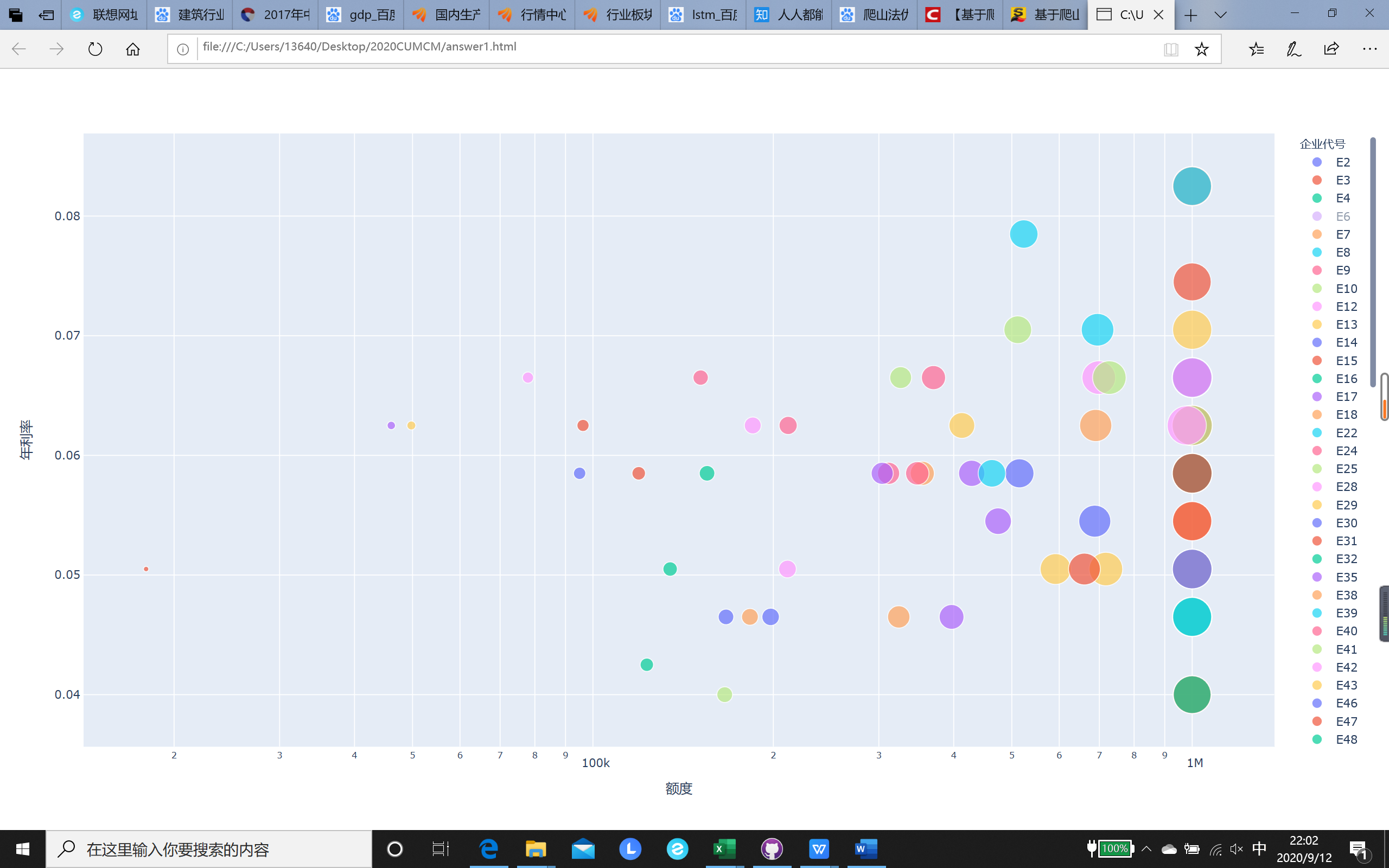




图 9 123家企业贷款额度、利率和

求得使得银行收益最高的每个企业的贷款额度、利率和当前值下的值。

## 4.2 问题二的分析与建模

### 4.2.1 问题分析

该问题要求给出总额为1亿元以内的信贷额度分配策略。结合问题一，为使用该风险回报模型，需要对附件2中的企业进行信誉评级和违约预测（填充样本缺失值）。通过附件2中302家企业的信贷风险进行量化分析，搭建神经网络模型学习银行对企业信誉评级的标准，以此评估无信贷企业信誉和违约概率。调用问题一中的模型，确定该银行在年度信贷总额为1亿元时对这些企业的信贷策略，并基于评价企业资金增长趋势调整企业信贷额度，可视化结果数据。

### 4.2.2 基于神经网络的信誉和违约预测模型建立

相较于问题一，问题二数据中的企业无信贷记录，这意味着银行无法进行贷款前信誉评价。企业的信誉和企业的实力强相关，因此可以通过问题一数据进行学习，建立企业资金流水和企业信誉的联系，预测企业的信誉评价。

我们建立基于神经网络的信誉预测模型。输入层为365×3个神经元，数据源为3年间每日企业资金存储，Mini-Batch设置为7（一周的天数）；隐含层设置6个神经元；输出层设置6个神经元，分别代表着信誉评价的“A”、“B”、“C”、“D”和是否违约“是”或“否”。在模型中定义：

* 一二层和二三层之间的网络参数，标准差为1，随机产生的数保持一致
* 神经网络前向传播过程
* 损失函数
* 反向传播算法

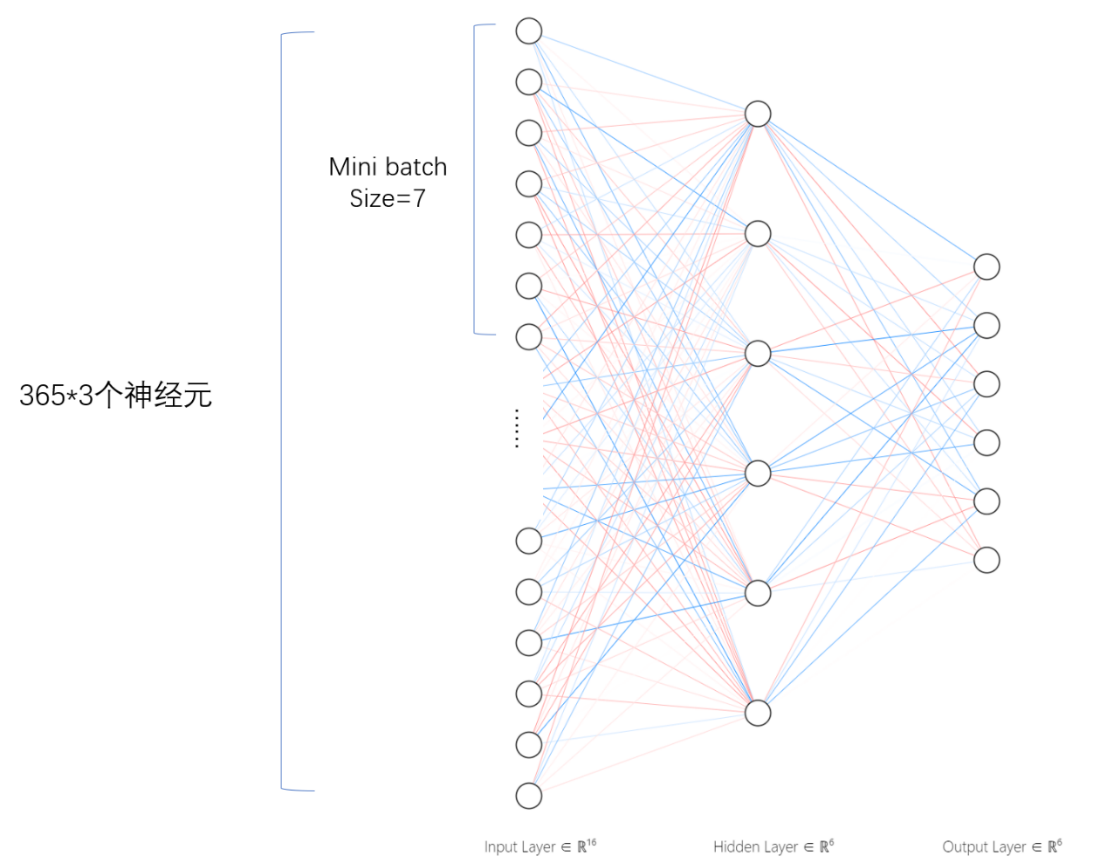


图 10预测信誉和违约的神经网络模型

通过选取的样本训练神经网络并更新参数，随着训练的进行，交叉熵逐渐变小，模型收敛。

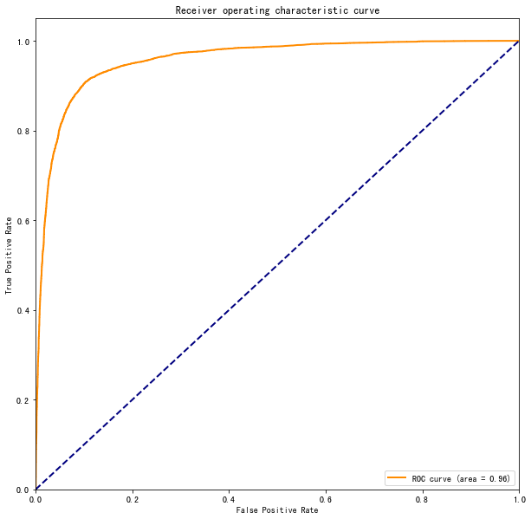


图 11神经网络模型训练收敛趋势

至此，我们能够预测302家无信贷记录企业的信誉评级。

将企业的信誉评级作为特征之一加入样本，利用问题一的模型计算每家企业的贷款额度、利率和该情况下银行收益。

### 4.2.3 有限总额度下基于评价资金趋势的减额策略

根据问题一的模型，可以统计302家企业各自的理论贷款额度。

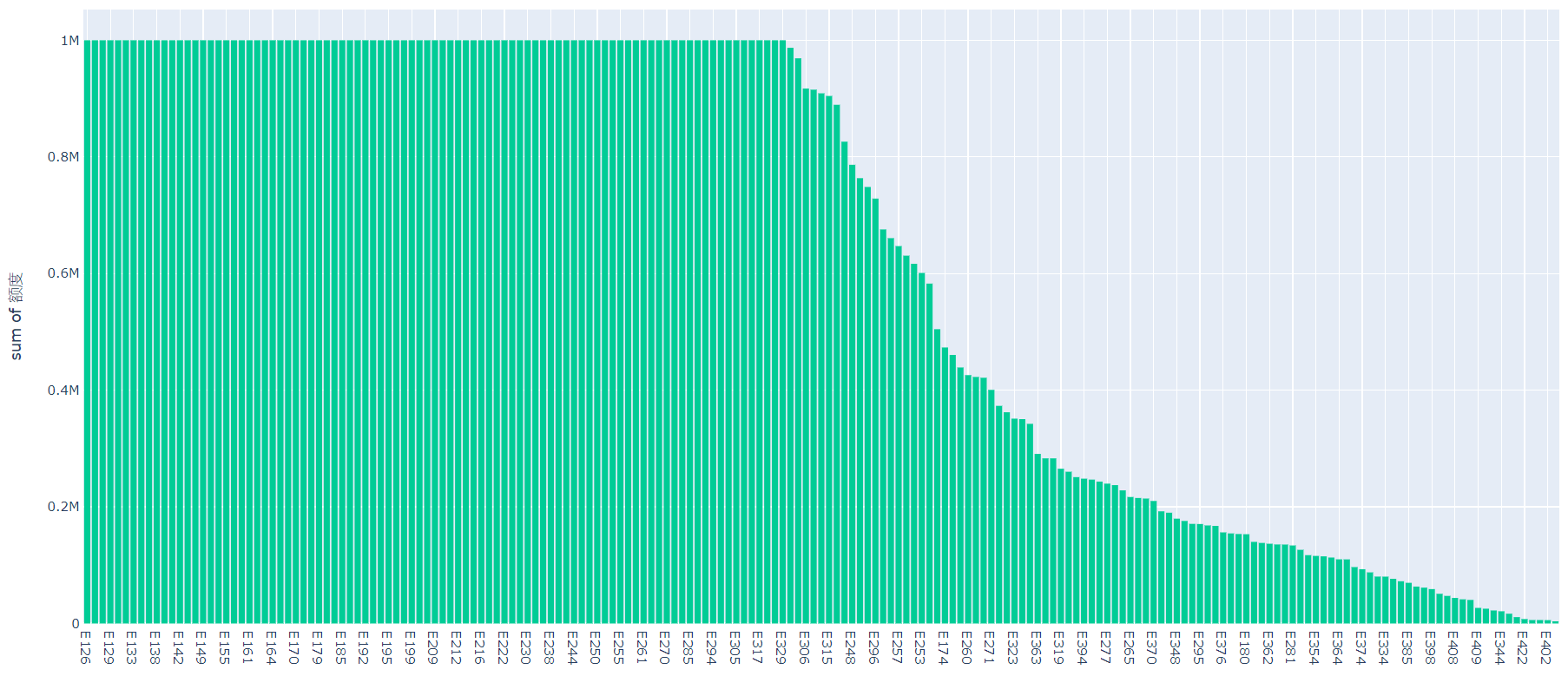


图 12 302家企业贷款额度分布

经统计，302家企业合计贷款额度119893799元，超出1亿元总额度，银行贷款额度不足时，需要在多个企业客户之间做出抉择，保留更可能高收益的客户，适当降低一些风险不够小的企业的额度。在企业储备资金均充足的情况下，银行更青睐增长趋势稳定、高速的企业。

我们需要描述企业的资金增长趋势（正/负）和速率快慢。定义为

其中，为企业资金增长随时间变化的斜率，描述了企业资金正/负增长，描述了企业资金增长/下降的速率。即：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 较大 | 较小 |
| + | | 快速增长 | 缓慢增长 |
| - | | 快速下降 | 缓慢下降 |

表 3资金增长期望值和方差意义

通过的实际意义，我们可以认为值越大，该企业越有潜力，是适合长期合作的伙伴，不应该局限于当前1年的贷款期限，未来长时间内能从该企业获得更高的利息收益；而值越小，即使该企业当前资金充足，但在未来的合作中收益比不上值大的企业。因此，在同一级别的企业中，我们优先保留值较大的企业的贷款额度，削减较小的企业的贷款额度。

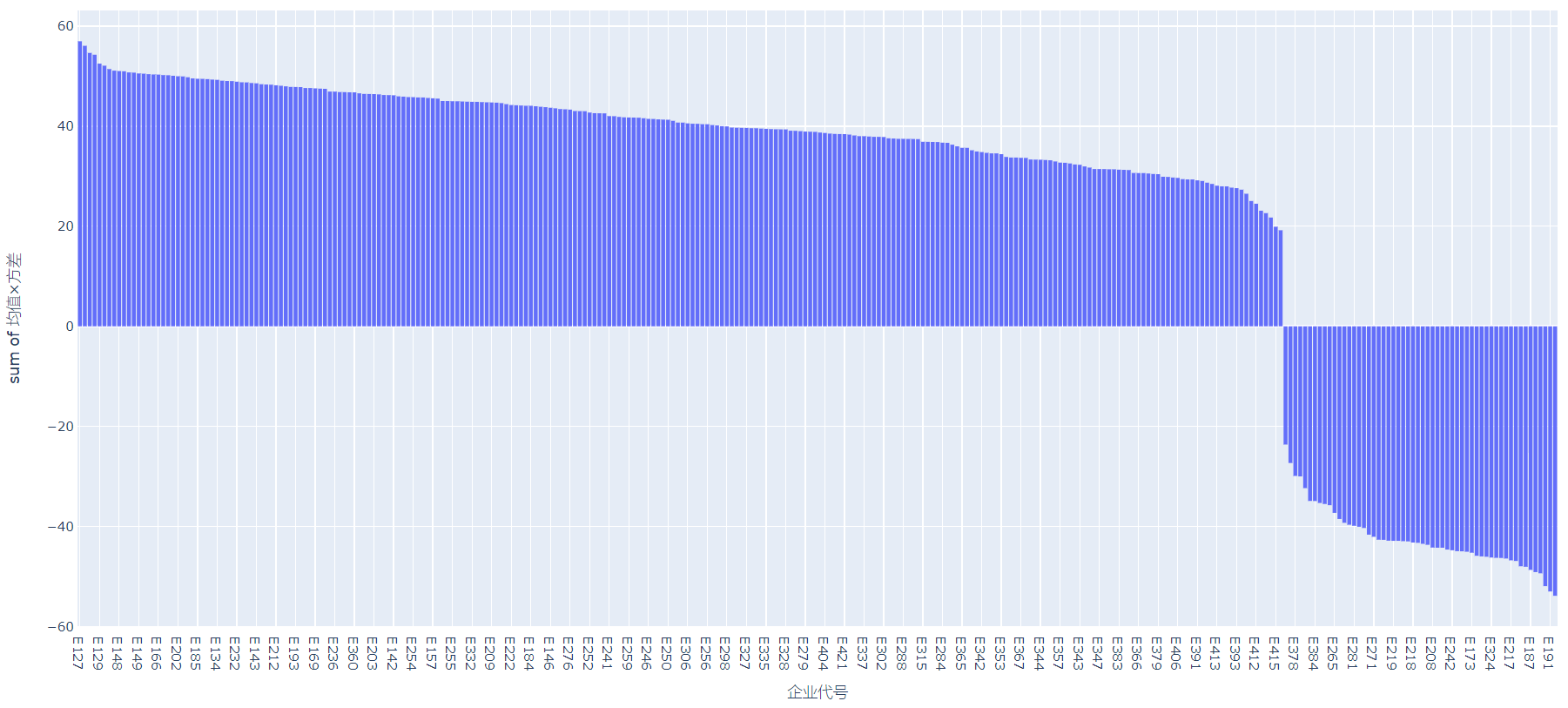


图 13 企业值分布图

通过削减策略，我们可以得到：基于问题一已有算法计算的放贷额度，在将总额控制在1亿元以内的情况下，银行放贷策略：

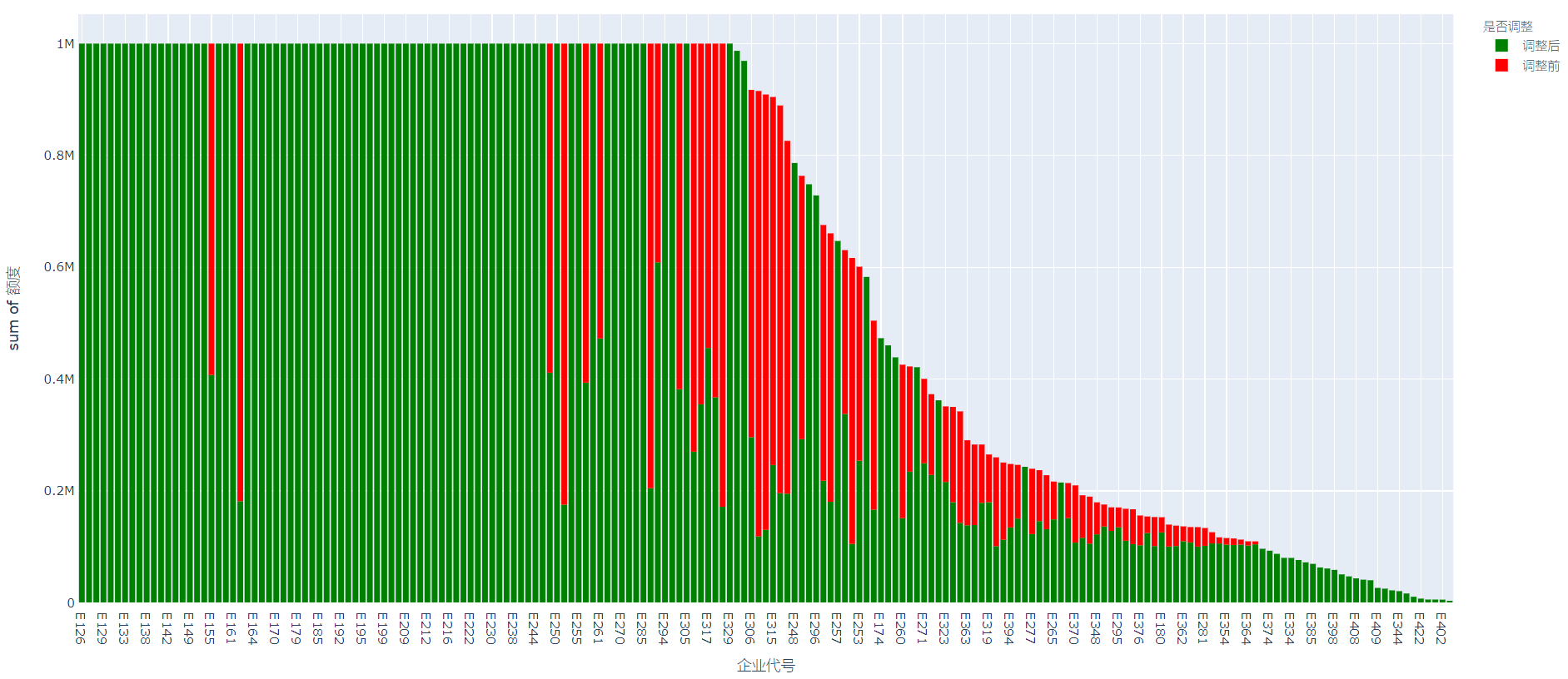


图 14 302家企业贷款额度削减前后对比

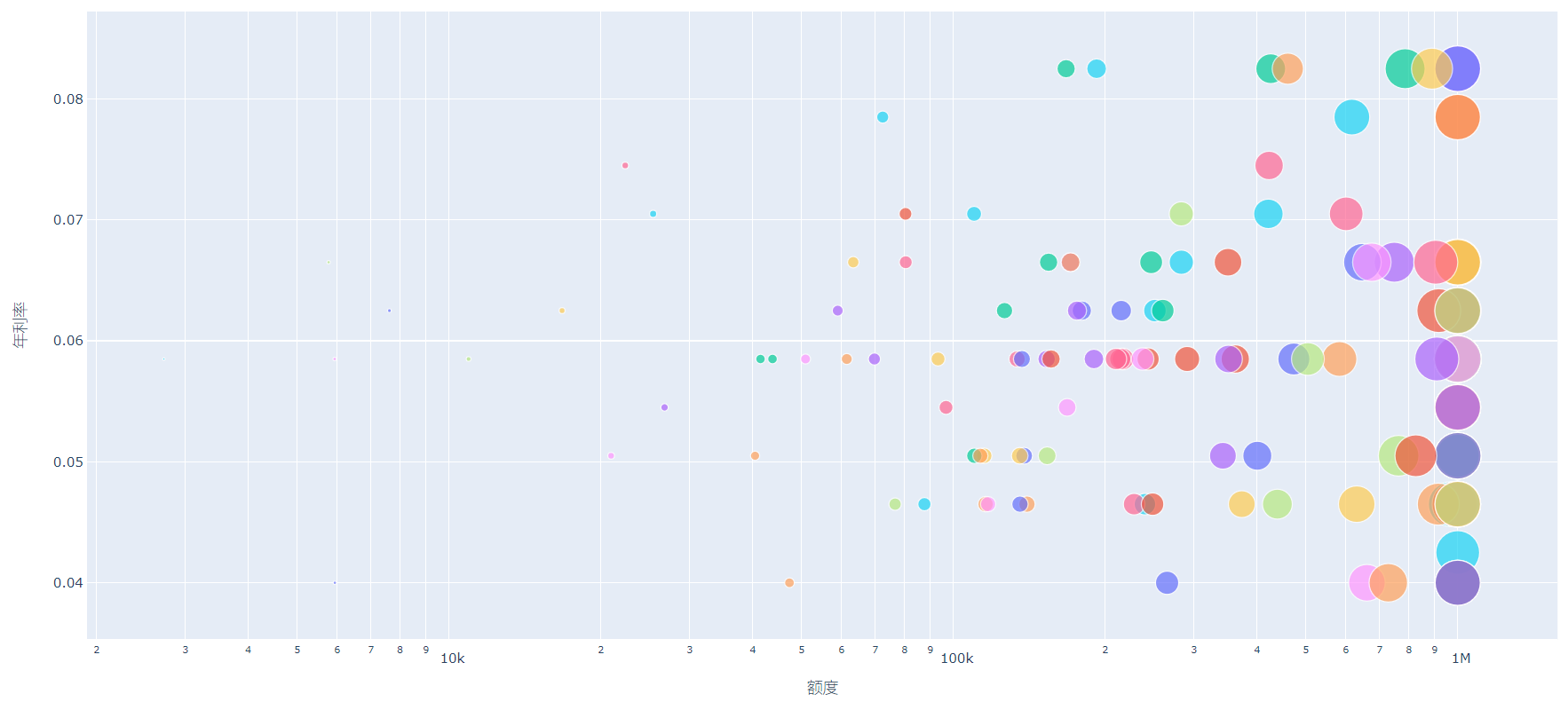


图 15 302家企业贷款额度、利率和

## 4.3 问题三的分析与建模

### 4.3.1 问题分析

突发因素对不同行业，不同类别企业会有不同的影响。给定数据的302家企业大致可分为八类：个体经营，商业贸易，服务业，工程建设，医药，高新技术，文体和日常生活；依据国务院颁布的《国家突发公共事件总体应急预案》，突发公共事件主要分为：自然灾害、事故灾难、公共卫生事件、社会安全事件。

为对企业分类，训练wiki语料数据库建立Word2Vec模型，构造词袋并设定聚类中心，依据词向量欧式距离将302家企业映射到相应的行业类别上。再对突发事件对各个行业的影响进行量化，建立影响矩阵和基于突发事件和行业两个自变量影响函数；最后给出在突发事件影响下银行相应的信贷调整策略。此外，利用已有数据验证量化模型的准确性，并给出选定条件下贷款额度削减结果。

### 4.3.2 聚类与词袋模型驱动的企业分类

首先对企业进行聚类，将企业人为地分为个体经营、商业贸易、服务业、工程建设等类。其次，为各个类寻找关键词，建立词袋列表如下：

|  |  |
| --- | --- |
| 企业类别 | 关键词 |
| 1. 个体经营 | 个体，经营 |
| 1. 商业贸易 | 商贸，批发，零售 |
| 1. 服务业 | 销售，咨询，劳务，物流，营销，房地产，酒店管理，包装，服务，事务所，招标 |
| 1. 工程建设 | 道路，桥梁，建筑，环境，装饰，园林 |
| 1. 医药 | 医疗，卫生，药品 |
| 1. 高新技术 | 科技，电子，电气，电器，邮电，通信，机械，网络 |
| 1. 文体 | 影视，广告，演艺 |
| 1. 日常生活 | 食品，服装，家居 |

表 4企业分类和关键词

接着使用Gensim中的Word2Vec模型，基于wiki语料库训练模型，再将每个词语转换成向量，找到词袋（bag of words）的中心。通过PCA降维得到二维向量，进行词向量可视化：

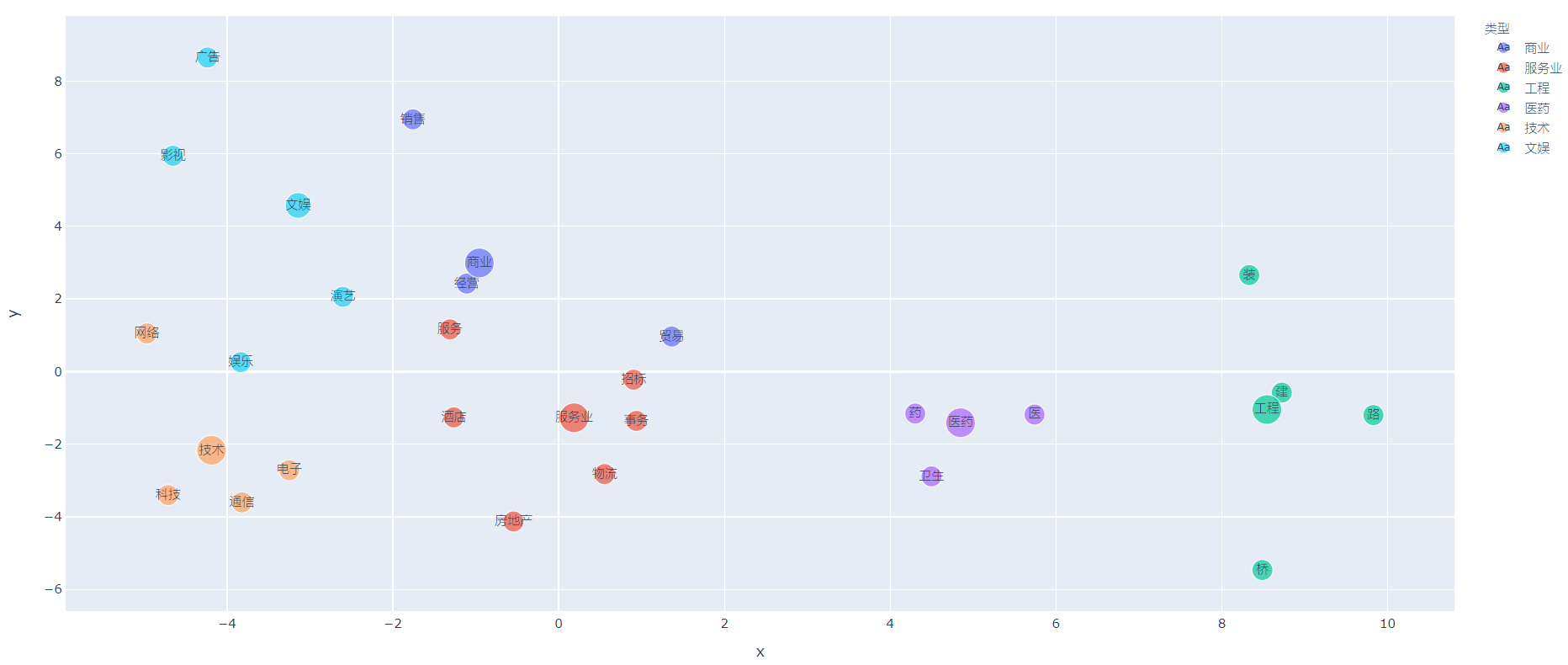


图 16行业聚类词袋可视化

计算企业关键词到每一行业词袋中心（认为是行业关键词中心）的欧式空间距离：

企业为指定一个行业的的概率与其距离成负相关，构造关于的函数得：

得出每个企业归类为N个类别的概率：

例如表示该企业为服务业的概率。

激活概率最高的类别，作为企业的归类。

激活后，能够将每一个企业对应到一个行业上，具体的分类结果为：



图 17企业分类后行业词云

由于现代企业具有交叉性，并考虑到模型的准确性，因此只在分类时对概率最高类别进行激活，在接下来的计算中保留维向量（即保留为其他类别的概率），以维向量作为量化后的影响的权重。

### 4.3.3 突发事件分类及其对行业影响量化

突发事件可以分为四类：自然灾害，事故灾难，公共卫生事件，社会安全事件。对于每一类事件，有一个维向量，第维值表示在该类事件下，第类企业（）所受的影响。例如，公共卫生事件对于不同企业影响的矩阵为：

该数据的计算基于公共卫生事件发生前后的行业增长数据。

对于任意一类行业，可以基于时间预测模型LSTM、通过事件发生前的数据预测“若事件不发生，该行业的增长值”，以事件发生时间内的预测值和实际值之差来量化突发事件对企业的影响。某一类突发事件对某一个行业的影响基准值可以定义为：

此外，同一类突发事件有着不同的影响程度，例如新冠病毒疫情的影响远高于一般的流感。因此，设置常数（定义指定突发事件时间内GDP影响），衡量特定一次突发事件在该类突发事件中对某一行业影响的加权值可以定义为：

以此来评价“指定突发事件对一个行业的影响”。

### 4.3.4 特定突发事件对特定企业影响函数

企业通过有权映射对应到一个归类的行业（图示为最大值激活后情况），行业和突发事件作为影响函数的参数进行计算，得出突发事件对企业的影响程度。



图 18影响函数与企业、行业和突发事件关系图

综合以上分析，可以定义特定突发事件对特定企业影响函数为：

即：以该企业为每一行业的概率为权重，突发事件对每一行业的影响为值，求该企业收到突发事件影响的加权平均值。

### 4.3.5 面对突发事件的银行放贷调整策略

依据上述算法，银行能够在突发事件发生时第一时间对企业受影响的情况进行评估，得出预估的。与问题二中“评估增长趋势的稳定和速度来评价企业潜力，削减潜力差的企业的资金”类似，在本问题中，银行“评估在突发事件中企业受影响程度，削减负面影响程度大的企业的资金”。

首先通过问题一的模型确定无限总额度下各企业的贷款额度分配；再评估突发事件中企业受影响程度，削减影响程度较大的企业的贷款额度；并且在受影响程度一定的情况下，优先削减增长潜力差的企业。

在突发事件中削减额度时，影响程度优先级高于企业增长潜力。其原因在于，突发事件在短期会强烈影响企业资金运转的情况下，即使潜力再大，在一定的贷款期限内也不一定能保持其势头；甚至在长远的角度，突发事件可能影响企业的潜力（例如企业设施破坏、外在因素导致大量人员流失、自然灾害后的地方政策）。

### 4.3.6 放贷策略和影响函数的验证：以2020新冠疫情为例

根据历史上疫情的GDP影响情况和2020新冠疫情产业数据，计算出对于不同企业影响的矩阵。选定企业“通讯器材有限公司”，根据上述算法得出：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 距离 | 概率 | 行业影响 | 加权平均值 |
| 个体经营 | 0.400 | 0.118 | -1.211 | -4.679 |
| 商业贸易 | 0.028 | 0.457 | -4.297 |
| 服务业 | 0.507 | 0.087 | -0.977 |
| 工程建设 | 0.880 | 0.016 | -0.119 |
| 医药 | 0.612 | 0.063 | 1.544 |
| 高新技术 | 0.389 | 0.121 | 1.564 |
| 文体 | 0.691 | 0.047 | -0.715 |
| 日常生活 | 0.496 | 0.090 | -0.468 |

表 5新冠疫情下企业影响值算法实例

经过验证，该模型能够较好地描述该企业地受影响程度。对其余企业进行抽样验证，均符合实际情况，因此该模型可信度较高。

通过上述放贷削减策略，可以将302家企业的贷款额度进行削减，前后对比如下：

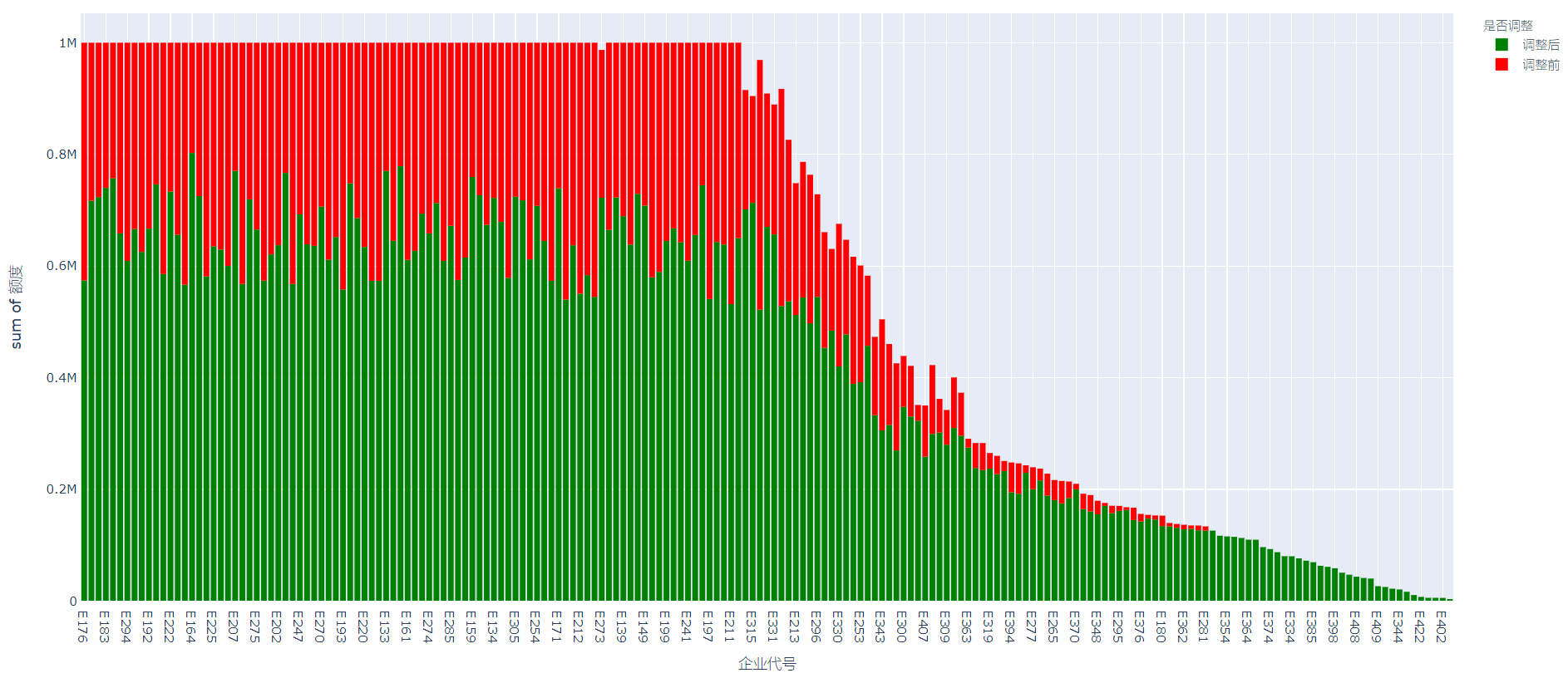


图 19 302家企业疫情期间削减贷款额度对比

# 五．模型的评价和改进

## 问题一

问题一中风险调整资本回报率模型充分考虑了风险与收益的关系，并将风险作为重要的因素引入建立模型。模型的参数即为问题中需要求解的值，通过函数的最大化直接能得出所需答案。因此问题一的模型非常契合题目，且符合事实依据，很好地结合了风险与收益。

模型中公式的建立符合逻辑，但在实际的计算和检验中存在一定的不足，且模型将风险优先于利润，导致预期收入偏低，是风格较为保守的模型，对于不同的银行来说不一定具有普适性。可以考虑添加对于银行承担风险的能力，来决定银行放贷的策略风格。

## 问题二

问题二中神经网络模型的信誉和违约预测是由企业资金数据驱动的，将数据一作为训练集充分利用了已有数据，对数据二的缺失值（无信贷→无信誉评价、无违约记录）进行了补充，进而能供利用问题一的模型进行计算，充分利用已有模型。

但在对于信誉和违约的预测，神经网络模型受限于数据量不足和特征处理不够合理，预测准确率没有达到接近完美的地步，有待改善。

## 问题三

问题三中企业分类的方法运用了自然语言处理的著名模型Word2Vec，依据词向量的欧氏空间距离判断词的相似性，计算企业分类的概率；并将突发事件进行了分类，设置了参数，提高了模型灵活度，能够很好的应对各种程度、各种类型的突发事件，为银行提供参考；额度调整策略主要基于贪心思想和预测。以上的思路在今年新冠疫情的历史事件下的数据进行验证，结果合理可靠。

其中的模型超参数较多，随着时代的发展、地理和文化的不同，突发事件的常量也会发生变化，该模型暂时不能完美地适应各种情况。如果能获得更好地数据，在超参数上进行动态地预测，便能有更好的模型。

# 六．参考文献

[1]周新辉,李昱喆,李富有.新冠疫情对中小服务型企业影响评估及对策研究——基于回归算法优化模型的分析预测[J].经济评论,2020(03):101-117.

[2]张伟如. 中国商业银行对小微企业信贷融资问题研究[D].对外经济贸易大学,2014.

[3]杨振. 我国商业银行中小企业贷款定价的实证分析[D].郑州大学,2017.

[4]Lee Dong Joo. 商业银行的风险调整绩效评估方法研究[D].吉林大学,2016.

[5]孙洪波.疫情背景下的中小企业危机管理对策[J].经济管理文摘,2020(17):75-76.

[6]刘焌懿. 浅谈疫情期间农商行信贷业务渠道建设[N]. 中国县域经济报,2020-03-05(007).

# 七．附录

## 问题一解

银行放贷123家企业（不放贷不列出）贷款额度、年利率和预期收益

|  |  |  |  |
| --- | --- | --- | --- |
| 企业代号 | 额度 | 年利率 | 预期收益 |
| E2 | 1000000 | 0.0665 | 33182.3 |
| E3 | 1000000 | 0.0465 | 35310.05 |
| E4 | 1000000 | 0.0665 | 37264.97 |
| E6 | 1000000 | 0.04 | 34800 |
| E7 | 1000000 | 0.0545 | 34358.92 |
| E8 | 1000000 | 0.0625 | 32918.9 |
| E9 | 1000000 | 0.0665 | 33771.13 |
| E10 | 1000000 | 0.0665 | 36315.35 |
| E12 | 1000000 | 0.0585 | 34878.14 |
| E13 | 1000000 | 0.04 | 31680 |
| E14 | 1000000 | 0.0665 | 36615.23 |
| E15 | 1000000 | 0.04 | 33760 |
| E16 | 1000000 | 0.0505 | 35084.76 |
| E17 | 1000000 | 0.0465 | 35666.38 |
| E18 | 1000000 | 0.0465 | 32952.99 |
| E22 | 1000000 | 0.0585 | 33345.97 |
| E24 | 1000000 | 0.0465 | 35666.38 |
| E25 | 1000000 | 0.0505 | 36186.2 |
| E28 | 1000000 | 0.0665 | 35682.16 |
| E29 | 1000000 | 0.0585 | 37015.93 |
| E30 | 1000000 | 0.0585 | 35616.04 |
| E31 | 1000000 | 0.0745 | 33271.34 |
| E32 | 1000000 | 0.0585 | 36353.93 |
| E35 | 46077.78 | 0.0625 | 1628.694 |
| E38 | 1000000 | 0.0625 | 36015.78 |
| E39 | 1000000 | 0.0505 | 36186.2 |
| E40 | 1000000 | 0.0625 | 37188.14 |
| E41 | 1000000 | 0.0625 | 37188.14 |
| E42 | 1000000 | 0.0505 | 35084.76 |
| E43 | 1000000 | 0.0505 | 35900.27 |
| E46 | 514903.6 | 0.0585 | 19446.5 |
| E47 | 119228.8 | 0.0585 | 4234.204 |
| E48 | 1000000 | 0.04 | 32720 |
| E49 | 1000000 | 0.0825 | 35703.45 |
| E50 | 1000000 | 0.0825 | 35703.45 |
| E51 | 1000000 | 0.0825 | 34316.72 |
| E53 | 151262 | 0.0665 | 5440.21 |
| E54 | 1000000 | 0.0465 | 34761.92 |
| E55 | 698460 | 0.0665 | 26028.09 |
| E58 | 1000000 | 0.0705 | 35290.71 |
| E59 | 687827.3 | 0.0545 | 23633 |
| E60 | 1000000 | 0.0545 | 35600.72 |
| E61 | 1000000 | 0.0465 | 35694.24 |
| E62 | 396727.3 | 0.0465 | 14160.88 |
| E63 | 690270.4 | 0.0625 | 23936.85 |
| E64 | 1000000 | 0.0465 | 34761.92 |
| E65 | 370103.1 | 0.0665 | 13440.42 |
| E66 | 727388.2 | 0.0665 | 25954.79 |
| E67 | 979414.1 | 0.0625 | 35274.36 |
| E68 | 591616.6 | 0.0505 | 21917.2 |
| E69 | 1000000 | 0.0505 | 36186.2 |
| E70 | 96250.74 | 0.0625 | 3402.14 |
| E71 | 134534 | 0.0505 | 4829.807 |
| E72 | 428608.2 | 0.0585 | 15865.33 |
| E73 | 354472.5 | 0.0585 | 13387.46 |
| E74 | 695179.5 | 0.0705 | 24533.38 |
| E75 | 347733.8 | 0.0585 | 12610.42 |
| E76 | 511630.4 | 0.0705 | 17759.11 |
| E77 | 184778.8 | 0.0625 | 6616.241 |
| E78 | 412743.1 | 0.0625 | 15349.15 |
| E79 | 197933.2 | 0.0465 | 7065.074 |
| E80 | 1000000 | 0.0585 | 36264.59 |
| E81 | 122998.1 | 0.0425 | 4152.547 |
| E84 | 474402.4 | 0.0545 | 16299.95 |
| E85 | 323851.6 | 0.0465 | 11559.64 |
| E86 | 523635 | 0.0785 | 18617.89 |
| E87 | 311414.8 | 0.0585 | 11293.33 |
| E88 | 326264.9 | 0.0665 | 11018.34 |
| E89 | 211171.2 | 0.0505 | 7236.88 |
| E90 | 717828.6 | 0.0505 | 25975.49 |
| E91 | 166730.9 | 0.0465 | 5645.085 |
| E92 | 661006.3 | 0.0505 | 23350.78 |
| E93 | 155023.5 | 0.0585 | 5635.712 |
| E94 | 303878.4 | 0.0585 | 11248.34 |
| E95 | 182802.9 | 0.0465 | 6525.011 |
| E96 | 463326 | 0.0585 | 17498.56 |
| E97 | 211705.3 | 0.0625 | 7624.732 |
| E98 | 165914.3 | 0.04 | 5773.816 |
| E104 | 77909.91 | 0.0665 | 2903.311 |
| E105 | 49768.79 | 0.0625 | 1850.809 |
| E106 | 95005.16 | 0.0585 | 3523.915 |
| E110 | 17961.02 | 0.0505 | 665.3893 |

## 数据一代码

|  |
| --- |
| 企业、发票数据对象设计Enterprise.py Invoice.py |
| from Invoice import Invoice  class Enterprise:  number: str  name: str  credit\_rating: str  break\_contract: bool  invoice\_list: ["Invoice"]  def \_\_init\_\_(  self, number: str, name: str, credit\_rating: str, break\_contract: bool  ):  self.number = number  self.name = name  self.credit\_rating = credit\_rating  self.break\_contract = break\_contract  self.invoice\_list = []  def add\_invoice(self, new\_invoice: "Invoice"):  self.invoice\_list.append(new\_invoice)  from datetime import date  class Invoice:  number: int  date: "date"  self\_enterprise: "Enterprise"  partner: str  amount: float  tax: float  sum\_money: float  state\_avaliable: bool  buy\_in: bool  def \_\_init\_\_(  self,  number: int,  date: "date",  self\_enterprise: "Enterprise",  partner: str,  amount: float,  tax: float,  sum\_money: float,  state\_avaliable: bool,  buy\_in: bool,  ):  self.number = number  self.date = date  self.self\_enterprise = self\_enterprise  self.partner = partner  self.amount = amount  self.tax = tax  self.sum\_money = sum\_money  self.state\_avaliable = state\_avaliable  self.buy\_in = buy\_in |

|  |
| --- |
| 问题一数据处理和建模 |
| import numpy as np  import pandas as pd  import plotly\_express as px  import matplotlib.pyplot as plt  from Invoice import Invoice  from Enterprise import Enterprise  # 读入数据  enterprise\_info=pd.read\_csv('./C/1\_info.csv')  N\_enterprise=enterprise\_info.shape[0]  # 将数据写进对象  enterprise\_dic={} # 企业代号 到 对象 的字典  for i in range(N\_enterprise):  number = enterprise\_info['企业代号'][i]  name = enterprise\_info['企业名称'][i]  credit\_rating = enterprise\_info['信誉评级'][i]  break\_contract\_str = enterprise\_info['是否违约'][i]  break\_contract = True if break\_contract\_str=='是' else False  enterprise\_object=Enterprise(number,name,credit\_rating,break\_contract)  enterprise\_dic[number]=enterprise\_object  # 将数据中日期的格式转换为date可识别的格式  from datetime import date  def get\_date(date\_str):  y,m,d=date\_str.split('/')  if len(m) != 2:  m = '0' + m  if len(d) != 2:  d = '0' + d  return date.fromisoformat(y+'-'+m+'-'+d)  # 将入账发票信息写入对象  invoice\_in = pd.read\_csv('./C/1\_in.csv')  N\_in = invoice\_in.shape[0]  for i in range(N\_in):  enterprise\_name = invoice\_in['企业代号'][i]  number = invoice\_in['发票号码'][i]  date\_str = invoice\_in['开票日期'][i]  date = get\_date(date\_str)  self\_enterprise = enterprise\_dic[enterprise\_name]  partner = invoice\_in['销方单位代号'][i]  amount = invoice\_in['金额'][i]  tax = invoice\_in['税额'][i]  sum\_money = invoice\_in['价税合计'][i]  state\_available\_str = invoice\_in['发票状态'][i]  state\_available = True if state\_available\_str == '有效发票' else False  invoice\_object = Invoice(number,date,self\_enterprise,partner,amount,tax,sum\_money,state\_available,True)  self\_enterprise.add\_invoice(invoice\_object)  # 将销账发票信息写入对象  invoice\_out = pd.read\_csv('./C/1\_out.csv')  N\_out = invoice\_out.shape[0]  for i in range(N\_out):  enterprise\_name = invoice\_out['企业代号'][i]  number = invoice\_out['发票号码'][i]  date\_str = invoice\_out['开票日期'][i]  date = get\_date(date\_str)  self\_enterprise = enterprise\_dic[enterprise\_name]  partner = invoice\_out['购方单位代号'][i]  amount = invoice\_out['金额'][i]  tax = invoice\_out['税额'][i]  sum\_money = invoice\_out['价税合计'][i]  state\_available\_str = invoice\_out['发票状态'][i]  state\_available = True if state\_available\_str == '有效发票' else False  invoice\_object = Invoice(number,date,self\_enterprise,partner,amount,tax,sum\_money,state\_available,False)  self\_enterprise.add\_invoice(invoice\_object)  # 统计每个企业的客户关系  invoice\_out = invoice\_out.sort\_values(by=['企业代号','购方单位代号'],axis=0,ascending=[True,True]).reset\_index(drop=True)  for enterprise in enterprise\_dic.values():  frame8 = pd.DataFrame(columns=['企业代号','购方单位代号','金额','税额','价税合计','交易数'])  temp = invoice\_out[invoice\_out.企业代号 == enterprise.number].reset\_index(drop=True)  N = temp.shape[0]  current\_buyer = temp['购方单位代号'][0]  amount\_sum = 0  tax\_sum = 0  sum\_sum = 0  nums\_sum = 0  for i in range(N):  temp1 = i  if temp['发票状态'][i] == '作废发票':  continue  if temp['购方单位代号'][i] != current\_buyer:  frame8 = frame8.append([{'企业代号':enterprise.number,'购方单位代号':temp['购方单位代号'][i],'金额':amount\_sum,'税额':tax\_sum,'价税合计':sum\_sum,'交易数':nums\_sum}],ignore\_index=True)  current\_buyer = temp['购方单位代号'][i]  amount\_sum = 0  tax\_sum = 0  sum\_sum = 0  nums\_sum = 0  amount\_sum = amount\_sum + temp['金额'][i]  tax\_sum = tax\_sum + temp['税额'][i]  sum\_sum = sum\_sum + temp['价税合计'][i]  nums\_sum = nums\_sum + 1  frame8 = frame8.append([{'企业代号':enterprise.number,'购方单位代号':temp['购方单位代号'][i],'金额':amount\_sum,'税额':tax\_sum,'价税合计':sum\_sum,'交易数':nums\_sum}],ignore\_index=True)  frame8 = frame8.sort\_values(by=['金额','交易数'],axis=0,ascending=[False,False]).reset\_index(drop=True)  total\_income = sum(frame8['金额'])  total\_trades = sum(frame8['交易数'])  graph = px.histogram(frame8, x="购方单位代号", y="金额",title=enterprise.number+' 收入总额:'+str(total\_income)+' 总交易数:'+str(total\_trades))  graph.write\_html('./1\_供求关系/'+enterprise.number+'.html')  graph2 = px.pie(frame8,values='金额',names='购方单位代号',title=enterprise.number+' 收入总额:'+str(total\_income)+' 总交易数:'+str(total\_trades))  graph2.update\_traces(textposition='inside', textinfo='percent+label')  graph2.write\_html('./1\_供求关系/'+enterprise.number+'\_pie.html')  # 统计每个企业资金随时间的变化并画图  frame = pd.DataFrame(columns=['日期','资金','企业代号','信誉评级'])  for enterprise in enterprise\_dic.values():  sum = 0  current\_date = enterprise.invoice\_list[0].date  for invoice in enterprise.invoice\_list:  temp = invoice  if invoice.date!=current\_date:  frame = frame.append([{'日期':current\_date,'资金':sum,'企业代号':enterprise.number,'信誉评级':enterprise.credit\_rating}],ignore\_index=True)  current\_date = invoice.date  if invoice.buy\_in == True:  sum = sum - invoice.sum\_money  else:  sum = sum + invoice.amount  frame = frame.append([{'日期':temp.date,'资金':sum,'企业代号':enterprise.number,'信誉评级':enterprise.credit\_rating}],ignore\_index=True)  graph = px.line(frame, x="日期", y="资金",color='企业代号',category\_orders={"信誉评级": ["A","B", "C", "D"]}, render\_mode="auto")  graph.write\_html('./1\_graph/total.html')  # 根据数据预测未来一年企业资金走势  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import mean\_squared\_error, r2\_score  def predict(frame\_predict):  N=len(frame\_predict['日期'])  days=frame\_predict['日期'][len(frame\_predict['日期'])-1]  x\_data = np.array(frame\_predict['日期']).reshape(-1, 1)  y\_data = np.array(frame\_predict['资金']).reshape(-1, 1)  # 数据分割  from sklearn.model\_selection import train\_test\_split  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_data, y\_data, test\_size=0.1)  model = LinearRegression()  model.fit(x\_train, y\_train.astype("int"))  y\_pred = model.predict(x\_test)    return model.predict(np.array([days+365]).reshape(-1, 1))[0]-frame\_predict['资金'][N-1]  # 计算预计额度  As = {}  for enterprise in enterprise\_dic.values():  temp = frame1[frame1.企业代号 == enterprise.number].reset\_index(drop=True)  N = temp.shape[0]  days = temp['日期'][N-1]  min\_days = 365  min\_index = 0  for i in range(N):  if abs(days-temp['日期'][i]-365)<min\_days:  min\_days=abs(days-temp['日期'][i]-365)  min\_index = i  delta = temp['资金'][N-1] - temp['资金'][min\_index]  As[enterprise.number] = delta/3  for en\_num in As.keys():  if As[en\_num]>1000000:  As[en\_num]=1000000  if As[en\_num]<0:  As[en\_num]=0  # 遍历年利率，计算每个企业期望收益最大值  i=0  rates = list(data\_rate\_and\_loss["贷款年利率"])  for enterprise in enterprise\_dic.values():  if enterprise.credit\_rating == "D" or As[enterprise.number]==0:  continue  ans = 0  ans\_A = 0  ans\_r = 0  A = As[enterprise.number]  for r in rates:  ans\_ = calc(enterprise, A, r)  if (ans\_>ans):  ans = ans\_  ans\_A = A  ans\_r = r  print(enterprise.number, A, ans, ans\_r)  # 计算企业月资金增长率的期望和方差  k\_mean\_var\_dic = {}  for enterprise in enterprise\_dic.values():  temp = frame1[frame1.企业代号 == enterprise.number].reset\_index(drop=True)  N = temp.shape[0]  days = temp['日期'][N-1]  ks=[]  i=0  while temp['日期'][i] + 30 < days:  today = temp['日期'][i]  min\_index = i  min\_days = 1000  for j in range(i+1,N):  delta = temp['日期'][j] - temp['日期'][i] - 30  if abs(delta) < min\_days:  min\_days = abs(delta)  min\_index = j  if delta > 0:  break  ks.append((temp['资金'][min\_index] - temp['资金'][i])/(temp['日期'][min\_index] - temp['日期'][i]))  i=min\_index  k\_mean\_var\_dic[enterprise.number] = [np.mean(ks),np.var(ks)]  frame7 = pd.DataFrame.from\_dict(k\_mean\_var\_dic,orient='index',columns=['增长率均值','增长率方差'])  frame7 = frame7.reset\_index().rename(columns={'index':'企业代号'})  frame7.to\_csv("./C/1\_资金增长率均值方差.csv",index=False,sep=',',encoding='utf\_8\_sig') |

## 问题一爬山法代码

|  |
| --- |
| 爬山法，求解RAROC最值是的A和r climbing.py |
| import numpy as np  pointdata=np.random.randint(1,100,size=(1000,2))  print(pointdata)  score=np.random.randn(1000)  score=score.reshape(-1,1)  print(score)  finaldata=np.hstack((pointdata,score))  print(finaldata)  #实时输出坐标轴x,y  def output(x,y):  return -x\*\*2-y\*\*2-x\*y+4\*x  '''  def output(x,y):  score=finaldata(x,y)  return score  '''  def test(x0,y0,fanx,fany):  #定义初始值的大小  point\_x=x0  point\_y=y0  step=0.001  point\_history=[]  score\_history=[]  i=0  while (point\_x<fanx[1] and point\_x>fanx[0]) and (point\_y<fany[1] and point\_y>fany[0]) and i<1e4:  x0=point\_x  y0=point\_y  while point\_x<fanx[1] and point\_x>fanx[0]:  last\_point\_x=point\_x  gradient = (output(point\_x+step, point\_y) - output(point\_x, point\_y)) / step  point\_x=point\_x+step\*np.sign(gradient)  if abs(output(point\_x,point\_y)-output(last\_point\_x,point\_y))<0.0001:  print(point\_x)  point\_history.append((point\_x,point\_y))  score\_history.append(output(point\_x,point\_y))  break  while point\_y<fany[1] and point\_y>fany[0]:  last\_point\_y = point\_y  gradient = (output(point\_x, point\_y+step) - output(point\_x, point\_y)) / step  point\_y = point\_y + step \* np.sign(gradient)  if abs(output(point\_x,point\_y)-output(point\_x,last\_point\_y))<0.00001:  point\_history.append((point\_x, point\_y))  score\_history.append(output(point\_x, point\_y))  print(point\_y)  break  if abs(output(point\_x, point\_y) - output(x0, y0)) < 0.0000001 or abs(point\_x-x0) < 0.00001:  break  i=i+1  print(point\_x, point\_y,x0, y0)  test(2,8,[-10,10],[-10,10])  # 画出图像如下进行检查  from mpl\_toolkits.mplot3d import Axes3D  import numpy as np  from matplotlib import pyplot as plt  fig = plt.figure(figsize=(10,6))  ax = Axes3D(fig)  x = np.arange(-10, 10, 0.5)  y = np.arange(-10, 10, 0.5)  X, Y = np.meshgrid(x, y)  Z =-X\*\*2 - Y\*\*2-X\*Y+4\*X  plt.xlabel('x')  plt.ylabel('y')  ax.plot\_surface(X, Y, Z, rstride=1, cstride=1, cmap='rainbow')  plt.show()  print(Z)  print(np.max(Z)) |

## 问题二解

银行放贷302家企业（不放贷不列出）贷款额度、年利率和预期收益、、调整后额度

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 企业代号 | 额度 | 年利率 | 预期收益 |  | 调整后额度 |
| E126 | 1000000 | 0.0465 | 33857.46 | 54.65818 | 1000000 |
| E127 | 1000000 | 0.0785 | 35555.08 | 56.98789 | 1000000 |
| E128 | 1000000 | 0.0585 | 35616.04 | 54.2986 | 1000000 |
| E129 | 1000000 | 0.0825 | 34805.47 | 52.51601 | 1000000 |
| E131 | 1000000 | 0.0585 | 37091.83 | 49.32635 | 1000000 |
| E132 | 1000000 | 0.0825 | 34805.47 | 51.11574 | 1000000 |
| E133 | 1000000 | 0.0465 | 33857.46 | 50.98437 | 1000000 |
| E134 | 1000000 | 0.0505 | 35084.76 | 49.27417 | 1000000 |
| E135 | 1000000 | 0.0825 | 36211.95 | 49.79209 | 1000000 |
| E138 | 1000000 | 0.0545 | 35488.91 | 48.34041 | 1000000 |
| E139 | 1000000 | 0.0665 | 35965.49 | 49.46803 | 1000000 |
| E140 | 1000000 | 0.0585 | 34878.14 | 50.71403 | 1000000 |
| E142 | 1000000 | 0.0625 | 35346.64 | 46.18745 | 1000000 |
| E144 | 1000000 | 0.0465 | 34761.92 | 50.75941 | 1000000 |
| E148 | 1000000 | 0.0665 | 37264.97 | 51.02544 | 1000000 |
| E149 | 1000000 | 0.0665 | 35965.49 | 50.55125 | 1000000 |
| E151 | 1000000 | 0.0585 | 35513.26 | 50.50825 | 1000000 |
| E152 | 1000000 | 0.0625 | 35346.64 | 46.92715 | 1000000 |
| E155 | 1000000 | 0.0545 | 34358.92 | -46.0566 | 100000 |
| E156 | 1000000 | 0.0465 | 35694.24 | 52.10441 | 1000000 |
| E159 | 1000000 | 0.0545 | 34358.92 | 47.63483 | 1000000 |
| E161 | 1000000 | 0.0665 | 35965.49 | 51.42865 | 1000000 |
| E162 | 1000000 | 0.0505 | 35084.76 | -47.9612 | 100000 |
| E163 | 1000000 | 0.0785 | 35030.57 | 49.01812 | 1000000 |
| E164 | 1000000 | 0.0585 | 37015.93 | 49.0434 | 1000000 |
| E165 | 1000000 | 0.0545 | 35600.72 | 46.44995 | 1000000 |
| E166 | 1000000 | 0.0585 | 37091.83 | 50.3279 | 1000000 |
| E170 | 1000000 | 0.0585 | 37091.83 | 47.63696 | 1000000 |
| E171 | 1000000 | 0.0505 | 35900.27 | 46.78276 | 1000000 |
| E176 | 1000000 | 0.0585 | 37091.83 | 45.82869 | 1000000 |
| E179 | 1000000 | 0.0465 | 35666.38 | 45.64811 | 1000000 |
| E183 | 1000000 | 0.0625 | 31672.52 | 46.36962 | 1000000 |
| E184 | 1000000 | 0.0425 | 32817.16 | 44.08774 | 1000000 |
| E185 | 1000000 | 0.0625 | 36497.21 | 49.48975 | 1000000 |
| E186 | 1000000 | 0.0625 | 33542.08 | 48.64155 | 1000000 |
| E190 | 1000000 | 0.0585 | 37015.93 | 50.35349 | 1000000 |
| E192 | 1000000 | 0.0825 | 35703.45 | 48.08485 | 1000000 |
| E193 | 1000000 | 0.04 | 34800 | 47.84704 | 1000000 |
| E194 | 1000000 | 0.0585 | 34727.7 | 47.8586 | 1000000 |
| E195 | 1000000 | 0.0665 | 36315.35 | 44.98879 | 1000000 |
| E196 | 1000000 | 0.0585 | 37767.26 | 44.71101 | 1000000 |
| E197 | 1000000 | 0.0465 | 33857.46 | 49.55831 | 1000000 |
| E199 | 1000000 | 0.0585 | 34727.7 | 50.41086 | 1000000 |
| E202 | 1000000 | 0.0585 | 33402.34 | 49.99011 | 1000000 |
| E207 | 1000000 | 0.0505 | 36186.2 | 50.08757 | 1000000 |
| E209 | 1000000 | 0.0625 | 36015.78 | 44.74895 | 1000000 |
| E210 | 1000000 | 0.0625 | 37188.14 | 48.30095 | 1000000 |
| E211 | 1000000 | 0.0585 | 37015.93 | 44.62409 | 1000000 |
| E212 | 1000000 | 0.0505 | 35900.27 | 48.18534 | 1000000 |
| E214 | 1000000 | 0.04 | 34800 | 46.21955 | 1000000 |
| E215 | 1000000 | 0.0505 | 37046.29 | 50.19798 | 1000000 |
| E216 | 1000000 | 0.0545 | 36265.76 | 45.52066 | 1000000 |
| E220 | 1000000 | 0.0665 | 35048.98 | 44.81881 | 1000000 |
| E221 | 1000000 | 0.0545 | 34838.12 | 45.97761 | 1000000 |
| E222 | 1000000 | 0.0505 | 36186.2 | 44.23507 | 1000000 |
| E225 | 1000000 | 0.0505 | 35900.27 | 45.04197 | 1000000 |
| E228 | 1000000 | 0.0585 | 34036.84 | 44.94639 | 1000000 |
| E230 | 1000000 | 0.0465 | 35666.38 | 47.51582 | 1000000 |
| E234 | 1000000 | 0.0505 | 37046.29 | 44.87685 | 1000000 |
| E235 | 1000000 | 0.0545 | 35600.72 | 43.38398 | 1000000 |
| E238 | 1000000 | 0.0585 | 37767.26 | 44.13783 | 1000000 |
| E241 | 1000000 | 0.0505 | 37046.29 | 42.02605 | 1000000 |
| E243 | 1000000 | 0.0585 | 35616.04 | 44.41346 | 1000000 |
| E244 | 1000000 | 0.0465 | 35666.38 | 43.58228 | 1000000 |
| E245 | 1000000 | 0.0585 | 37091.83 | 44.90879 | 1000000 |
| E247 | 1000000 | 0.0625 | 37188.14 | 40.40653 | 1000000 |
| E250 | 1000000 | 0.0585 | 36353.93 | 41.29686 | 1000000 |
| E251 | 1000000 | 0.0665 | 35048.98 | 38.87079 | 100000 |
| E254 | 1000000 | 0.0585 | 36353.93 | 45.80701 | 1000000 |
| E255 | 1000000 | 0.0585 | 34036.84 | 44.99602 | 1000000 |
| E256 | 1000000 | 0.0505 | 37046.29 | 40.40017 | 1000000 |
| E258 | 1000000 | 0.0465 | 36228.78 | 46.81892 | 1000000 |
| E261 | 1000000 | 0.0545 | 34358.92 | 37.45953 | 100000 |
| E266 | 1000000 | 0.0585 | 37767.26 | 43.42947 | 1000000 |
| E267 | 1000000 | 0.0465 | 35666.38 | 45.75084 | 1000000 |
| E270 | 1000000 | 0.0545 | 36265.76 | 41.59535 | 1000000 |
| E274 | 1000000 | 0.0505 | 31826.54 | 43.99138 | 1000000 |
| E275 | 1000000 | 0.0545 | 35600.72 | 42.58948 | 1000000 |
| E285 | 1000000 | 0.0505 | 37046.29 | 42.60966 | 1000000 |
| E289 | 1000000 | 0.0505 | 37046.29 | 39.6056 | 143050 |
| E290 | 1000000 | 0.0585 | 33345.97 | 40.72702 | 1000000 |
| E294 | 1000000 | 0.0585 | 37767.26 | 42.00336 | 1000000 |
| E301 | 1000000 | 0.0465 | 35310.05 | 40.73351 | 1000000 |
| E303 | 1000000 | 0.0665 | 35682.16 | 40.01482 | 1000000 |
| E305 | 1000000 | 0.04 | 34800 | 50.22381 | 1000000 |
| E311 | 1000000 | 0.0465 | 34761.92 | 39.69616 | 1000000 |
| E316 | 1000000 | 0.0585 | 37767.26 | 39.42296 | 100000 |
| E317 | 1000000 | 0.0585 | 37091.83 | 37.54443 | 100000 |
| E318 | 1000000 | 0.0465 | 35666.38 | 36.68229 | 100000 |
| E328 | 1000000 | 0.0625 | 35346.64 | 39.36474 | 100000 |
| E329 | 1000000 | 0.0625 | 36015.78 | 44.08938 | 1000000 |
| E273 | 987034 | 0.0465 | 35203.93 | 41.08197 | 987034 |
| E252 | 969027 | 0.0465 | 34588.68 | 42.73192 | 969027 |
| E306 | 917181 | 0.0625 | 32840.84 | 40.57078 | 917181 |
| E310 | 915238 | 0.0465 | 30159.83 | 38.52841 | 100000 |
| E338 | 908977 | 0.0585 | 33044.89 | 38.89756 | 100000 |
| E315 | 904459 | 0.0665 | 33116.98 | 36.89333 | 100000 |
| E331 | 889350 | 0.0825 | 28780.87 | 39.10757 | 100000 |
| E321 | 826061 | 0.0505 | 29892.01 | 38.46397 | 100000 |
| E248 | 786498 | 0.0825 | 27374.43 | 41.37163 | 786498 |
| E282 | 763520 | 0.0505 | 28285.59 | 39.59285 | 100000 |
| E213 | 748432 | 0.0665 | 27404.01 | 43.00253 | 748432 |
| E296 | 728516 | 0.04 | 25352.36 | 41.70019 | 728516 |
| E330 | 675619 | 0.0665 | 25176.92 | 36.30986 | 100000 |
| E284 | 660662 | 0.04 | 22991.04 | 36.71327 | 100000 |
| E257 | 647046 | 0.0665 | 24112.15 | 41.73469 | 647046 |
| E191 | 630655 | 0.0465 | 22847.86 | -52.9993 | 100000 |
| E237 | 616736 | 0.0785 | 21928.1 | 39.52755 | 100000 |
| E253 | 600993 | 0.0705 | 19466.9 | 40.23277 | 600993 |
| E276 | 582776 | 0.0585 | 20756.17 | 43.32132 | 582776 |
| E343 | 504569 | 0.0585 | 18343.07 | 32.29324 | 100000 |
| E174 | 473237 | 0.0585 | 17553.23 | 43.81291 | 473237 |
| E262 | 460433 | 0.0825 | 16673.18 | 41.44821 | 460433 |
| E300 | 438968 | 0.0465 | 15271.29 | 41.32359 | 438968 |
| E260 | 425848 | 0.0825 | 15204.24 | 38.15374 | 100000 |
| E298 | 422682 | 0.0745 | 13841.15 | 39.96768 | 422682 |
| E263 | 421317 | 0.0705 | 14897.59 | 42.57171 | 421317 |
| E271 | 400563 | 0.0505 | 14380.32 | -42.0681 | 100000 |
| E361 | 373165 | 0.0465 | 12296.9 | 35.68035 | 100000 |
| E160 | 362180 | 0.0585 | 13678.55 | 49.9538 | 362180 |
| E323 | 351215 | 0.0585 | 13027.21 | 37.41996 | 100000 |
| E407 | 350377 | 0.0665 | 13056.79 | 34.57356 | 100000 |
| E309 | 342319 | 0.0505 | 12289.34 | 39.36981 | 100000 |
| E363 | 290734 | 0.0585 | 10761.79 | 31.96846 | 100000 |
| E327 | 283139 | 0.0665 | 10103.01 | 39.66022 | 283139 |
| E357 | 283112 | 0.0705 | 10010.72 | 32.72664 | 100000 |
| E319 | 265334 | 0.04 | 9233.623 | 38.43883 | 100000 |
| E380 | 260170 | 0.0625 | 8726.644 | 39.08411 | 100000 |
| E341 | 250940 | 0.0625 | 8701.969 | 37.88419 | 100000 |
| E394 | 248308 | 0.0465 | 8638.405 | 31.28374 | 100000 |
| E350 | 246725 | 0.0665 | 8959.904 | 32.56038 | 100000 |
| E349 | 243269 | 0.0585 | 8448.173 | 44.19478 | 243269 |
| E277 | 239773 | 0.0465 | 7901.238 | 39.01249 | 100000 |
| E358 | 237240 | 0.0585 | 8624.607 | 33.24854 | 100000 |
| E356 | 228224 | 0.0465 | 7933.504 | 31.73171 | 100000 |
| E265 | 216894 | 0.0585 | 8044.996 | -37.2865 | 100000 |
| E332 | 215170 | 0.0625 | 7217.25 | 44.88285 | 215170 |
| E342 | 214229 | 0.0585 | 7629.988 | 34.83099 | 100000 |
| E370 | 210074 | 0.0585 | 7776.084 | 33.74272 | 100000 |
| E297 | 192320 | 0.0825 | 6693.788 | -46.2728 | 100000 |
| E365 | 189961 | 0.0585 | 6596.909 | 35.68649 | 100000 |
| E348 | 179743 | 0.0625 | 6435.929 | 30.63271 | 100000 |
| E351 | 175826 | 0.0625 | 6332.511 | 31.39332 | 100000 |
| E326 | 170806 | 0.0665 | 6032.141 | 35.98202 | 100000 |
| E295 | 170574 | 0.0665 | 6194.454 | 33.86126 | 100000 |
| E372 | 168139 | 0.0545 | 5405.823 | 30.45315 | 100000 |
| E308 | 167319 | 0.0825 | 5741.839 | 38.73663 | 100000 |
| E376 | 156187 | 0.0585 | 5781.407 | 31.43069 | 100000 |
| E335 | 154448 | 0.0665 | 5755.501 | 39.48853 | 100000 |
| E371 | 153438 | 0.0505 | 5508.465 | 31.44249 | 100000 |
| E180 | 153059 | 0.0585 | 5209.644 | 39.67926 | 153059 |
| E339 | 139921 | 0.0465 | 4610.816 | 34.57577 | 100000 |
| E375 | 138129 | 0.0505 | 4843.739 | 29.42261 | 100000 |
| E362 | 136749 | 0.0585 | 4971.364 | 36.86082 | 100000 |
| E406 | 135525 | 0.0465 | 4588.532 | 29.67515 | 100000 |
| E392 | 135374 | 0.0505 | 4749.564 | 30.55307 | 100000 |
| E281 | 133787 | 0.0585 | 4553.686 | -39.8663 | 100000 |
| E322 | 126297 | 0.0625 | 4609.488 | -35.7533 | 100000 |
| E391 | 117168 | 0.0465 | 3967.01 | 29.18614 | 100000 |
| E354 | 115869 | 0.0465 | 4135.856 | 32.95881 | 100000 |
| E347 | 115137 | 0.0505 | 4133.449 | 31.44108 | 100000 |
| E386 | 113109 | 0.0505 | 3968.402 | 38.32765 | 100000 |
| E364 | 110026 | 0.0505 | 4076.055 | 32.35393 | 100000 |
| E387 | 109907 | 0.0705 | 3878.696 | 27.7247 | 100000 |
| E388 | 96702 | 0.0545 | 3368.916 | 28.71603 | 96702 |
| E374 | 93243 | 0.0585 | 3381.42 | 33.32283 | 93243 |
| E368 | 87625 | 0.0465 | 3046.013 | 31.23765 | 87625 |
| E413 | 80457 | 0.0665 | 2921.824 | 28.11319 | 80457 |
| E334 | 80367 | 0.0705 | 2836.208 | 36.84138 | 80367 |
| E390 | 76640 | 0.0465 | 2664.153 | 43.0375 | 76640 |
| E355 | 72422 | 0.0785 | 2536.984 | 33.73967 | 72422 |
| E385 | 69733 | 0.0585 | 2535.069 | 32.69442 | 69733 |
| E405 | 63313 | 0.0665 | 2259.145 | 37.44804 | 63313 |
| E399 | 61465 | 0.0585 | 2092.074 | 29.74574 | 61465 |
| E398 | 58982 | 0.0625 | 2084.815 | 29.86774 | 58982 |
| E403 | 50916 | 0.0585 | 1733.02 | 31.38337 | 50916 |
| E366 | 47318 | 0.04 | 1646.666 | 30.63772 | 47318 |
| E408 | 43794 | 0.0585 | 1592.084 | 29.90416 | 43794 |
| E395 | 41454 | 0.0585 | 1565.604 | 33.35176 | 41454 |
| E410 | 40438 | 0.0505 | 1463.298 | 27.31814 | 40438 |
| E409 | 26751 | 0.0545 | 970.1453 | 25.05225 | 26751 |
| E400 | 25391 | 0.0705 | 868.3095 | 39.73471 | 25391 |
| E401 | 22356 | 0.0745 | 771.8628 | 27.98629 | 22356 |
| E344 | 20959 | 0.0505 | 776.4533 | 33.30649 | 20959 |
| E418 | 16755 | 0.0625 | 611.5108 | 29.05697 | 16755 |
| E416 | 10934 | 0.0585 | 412.9473 | 26.49884 | 10934 |
| E422 | 7617 | 0.0625 | 269.2353 | 19.20558 | 7617 |
| E393 | 5937 | 0.04 | 200.4331 | 27.62631 | 5937 |
| E417 | 5933 | 0.0585 | 219.6155 | 21.75169 | 5933 |
| E402 | 5771 | 0.0665 | 198.6136 | 35.19193 | 5771 |
| E412 | 2720 | 0.0585 | 98.6397 | 24.51094 | 2720 |

## 数据二代码

|  |
| --- |
| 问题二数据处理和模型建立 |
| import numpy as np  import pandas as pd  import plotly\_express as px  import matplotlib.pyplot as plt  from Invoice import Invoice  from Enterprise import Enterprise  # 计算企业月资金增长率的期望和方差  k\_mean\_var\_dic = {}  for enterprise in enterprise\_dic.values():  temp = frame1[frame1.企业代号 == enterprise.number].reset\_index(drop=True)  N = temp.shape[0]  days = temp['日期'][N-1]  ks=[]  i=0  while temp['日期'][i] + 30 < days:  today = temp['日期'][i]  min\_index = i  min\_days = 1000  for j in range(i+1,N):  delta = temp['日期'][j] - temp['日期'][i] - 30  if abs(delta) < min\_days:  min\_days = abs(delta)  min\_index = j  if delta > 0:  break  ks.append((temp['资金'][min\_index] - temp['资金'][i])/(temp['日期'][min\_index] - temp['日期'][i]))  i=min\_index  k\_mean\_var\_dic[enerprise.number] = [np.mean(ks),np.var(ks)]  frame7 = pd.DataFrame.from\_dict(k\_mean\_var\_dic,orient='index',columns=['增长率均值','增长率方差'])  frame7 = frame7.reset\_index().retname(columns={'index':'企业代号'})  frame7.to\_csv("./C/2\_资金增长率均值方差.csv",index=False,sep=',',encoding='utf\_8\_sig')  frame9 = pd.read\_csv('./C/2\_资金增长率均值方差.csv')  frame10 = pd.DataFrame(columns=['企业代号','均值×方差'])  for i in range(frame9.shape[0]):  frame10 = frame10.append([{'企业代号':frame9['企业代号'][i],'均值×方差':np.log2(frame9['增长率均值'][i]\*frame9['增长率方差'][i]) if frame9['增长率均值'][i]\*frame9['增长率方差'][i] > 0 else -np.log2(-frame9['增长率均值'][i]\*frame9['增长率方差'][i])}],ignore\_index=True)  frame10 = frame10.sort\_values(by=['均值×方差','企业代号'],axis=0,ascending=[False,True]).reset\_index(drop=True)  frame10.to\_csv("./C/2\_增长率均值×方差.csv",index=False,sep=',',encoding='utf\_8\_sig')  graph3 = px.histogram(frame10, x="企业代号", y="均值×方差",color\_discrete\_map=color\_dic)  graph3.write\_html('./2\_graph/增长率均值×方差.html') |

## 问题二神经网络代码

|  |
| --- |
| 神经网络模型训练和建立 nn.py |
| import tensorflow as tf  # 导入TensorFlow工具包并简称为tf  from numpy.random import RandomState  # 导入numpy工具包，生成模拟数据集  batch\_size = 7  # 定义训练数据batch的大小  w1 = tf.Variable(tf.random\_normal([2, 3], stddev=1, seed=1))  w2 = tf.Variable(tf.random\_normal([3, 1], stddev=1, seed=1))  # 分别定义一二层和二三层之间的网络参数，标准差为1，随机产生的数保持一致  x = tf.placeholder(tf.float32, shape=(None, 2), name="x-input")  y\_ = tf.placeholder(tf.float32, shape=(None, 1), name="y-input")  # 输入为两个维度，即两个特征，输出为一个标签,声明数据类型float32，None即一个batch大小  # y\_是真实的标签  a = tf.matmul(x, w1)  y = tf.matmul(a, w2)  # 定义神经网络前向传播过程  cross\_entropy = -tf.reduce\_mean(y\_ \* tf.log(tf.clip\_by\_value(y, 1e-10, 1.0)))  train\_step = tf.train.AdamOptimizer(0.001).minimize(cross\_entropy)  # 定义损失函数和反向传播算法  rdm = RandomState(1)  dataset\_size = 128  # 产生128组数据  X = rdm.rand(dataset\_size, 2)  Y = [[int(x1 + x2 < 1)] for (x1, x2) in X]  # 将所有x1+x2<1的样本视为正样本，表示为1；其余为0  # 创建会话来运行TensorFlow程序  with tf.Session() as sess:  init\_op = tf.global\_variables\_initializer()  # 初始化变量  sess.run(init\_op)  print(sess.run(w1))  print(sess.run(w2))  # 打印出训练网络之前网络参数的值  STEPS = 5000  # 设置训练的轮数  for i in range(STEPS):  start = (i \* batch\_size) % dataset\_size  end = min(start + batch\_size, dataset\_size)  # 每次选取batch\_size个样本进行训练  sess.run(train\_step, feed\_dict={x: X[start:end], y\_: Y[start:end]})  # 通过选取的样本训练神经网络并更新参数  if i % 1000 == 0:  total\_cross\_entropy = sess.run(cross\_entropy, feed\_dict={x: X, y\_: Y})  print(  "After %d training step(s),cross entropy on all data is %g"  % (i, total\_cross\_entropy)  )  # 每隔一段时间计算在所有数据上的交叉熵并输出，随着训练的进行，交叉熵逐渐变小  print(sess.run(w1))  print(sess.run(w2))  # 打印出训练之后神经网络参数的值 |

## 问题三解

银行放贷302家企业（不放贷不列出）贷款额度、年利率和预期收益、、调整后额度

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 企业代号 | 额度 | 预期收益 | 年利率 | 增长率均值×方差 | 调整后额度 |
| E126 | 1000000 | 33857.46 | 0.0465 | 54.65818 | 1000000 |
| E127 | 1000000 | 35555.08 | 0.0785 | 56.98789 | 1000000 |
| E128 | 1000000 | 35616.04 | 0.0585 | 54.2986 | 1000000 |
| E129 | 1000000 | 34805.47 | 0.0825 | 52.51601 | 1000000 |
| E131 | 1000000 | 37091.83 | 0.0585 | 49.32635 | 1000000 |
| E132 | 1000000 | 34805.47 | 0.0825 | 51.11574 | 1000000 |
| E133 | 1000000 | 33857.46 | 0.0465 | 50.98437 | 1000000 |
| E134 | 1000000 | 35084.76 | 0.0505 | 49.27417 | 1000000 |
| E135 | 1000000 | 36211.95 | 0.0825 | 49.79209 | 1000000 |
| E138 | 1000000 | 35488.91 | 0.0545 | 48.34041 | 1000000 |
| E139 | 1000000 | 35965.49 | 0.0665 | 49.46803 | 1000000 |
| E140 | 1000000 | 34878.14 | 0.0585 | 50.71403 | 1000000 |
| E142 | 1000000 | 35346.64 | 0.0625 | 46.18745 | 1000000 |
| E144 | 1000000 | 34761.92 | 0.0465 | 50.75941 | 1000000 |
| E148 | 1000000 | 37264.97 | 0.0665 | 51.02544 | 1000000 |
| E149 | 1000000 | 35965.49 | 0.0665 | 50.55125 | 1000000 |
| E151 | 1000000 | 35513.26 | 0.0585 | 50.50825 | 1000000 |
| E152 | 1000000 | 35346.64 | 0.0625 | 46.92715 | 1000000 |
| E155 | 1000000 | 34358.92 | 0.0545 | -46.0566 | 407499 |
| E156 | 1000000 | 35694.24 | 0.0465 | 52.10441 | 1000000 |
| E159 | 1000000 | 34358.92 | 0.0545 | 47.63483 | 1000000 |
| E161 | 1000000 | 35965.49 | 0.0665 | 51.42865 | 1000000 |
| E162 | 1000000 | 35084.76 | 0.0505 | -47.9612 | 181805 |
| E163 | 1000000 | 35030.57 | 0.0785 | 49.01812 | 1000000 |
| E164 | 1000000 | 37015.93 | 0.0585 | 49.0434 | 1000000 |
| E165 | 1000000 | 35600.72 | 0.0545 | 46.44995 | 1000000 |
| E166 | 1000000 | 37091.83 | 0.0585 | 50.3279 | 1000000 |
| E170 | 1000000 | 37091.83 | 0.0585 | 47.63696 | 1000000 |
| E171 | 1000000 | 35900.27 | 0.0505 | 46.78276 | 1000000 |
| E176 | 1000000 | 37091.83 | 0.0585 | 45.82869 | 1000000 |
| E179 | 1000000 | 35666.38 | 0.0465 | 45.64811 | 1000000 |
| E183 | 1000000 | 31672.52 | 0.0625 | 46.36962 | 1000000 |
| E184 | 1000000 | 32817.16 | 0.0425 | 44.08774 | 1000000 |
| E185 | 1000000 | 36497.21 | 0.0625 | 49.48975 | 1000000 |
| E186 | 1000000 | 33542.08 | 0.0625 | 48.64155 | 1000000 |
| E190 | 1000000 | 37015.93 | 0.0585 | 50.35349 | 1000000 |
| E192 | 1000000 | 35703.45 | 0.0825 | 48.08485 | 1000000 |
| E193 | 1000000 | 34800 | 0.04 | 47.84704 | 1000000 |
| E194 | 1000000 | 34727.7 | 0.0585 | 47.8586 | 1000000 |
| E195 | 1000000 | 36315.35 | 0.0665 | 44.98879 | 1000000 |
| E196 | 1000000 | 37767.26 | 0.0585 | 44.71101 | 1000000 |
| E197 | 1000000 | 33857.46 | 0.0465 | 49.55831 | 1000000 |
| E199 | 1000000 | 34727.7 | 0.0585 | 50.41086 | 1000000 |
| E202 | 1000000 | 33402.34 | 0.0585 | 49.99011 | 1000000 |
| E207 | 1000000 | 36186.2 | 0.0505 | 50.08757 | 1000000 |
| E209 | 1000000 | 36015.78 | 0.0625 | 44.74895 | 1000000 |
| E210 | 1000000 | 37188.14 | 0.0625 | 48.30095 | 1000000 |
| E211 | 1000000 | 37015.93 | 0.0585 | 44.62409 | 1000000 |
| E212 | 1000000 | 35900.27 | 0.0505 | 48.18534 | 1000000 |
| E214 | 1000000 | 34800 | 0.04 | 46.21955 | 1000000 |
| E215 | 1000000 | 37046.29 | 0.0505 | 50.19798 | 1000000 |
| E216 | 1000000 | 36265.76 | 0.0545 | 45.52066 | 1000000 |
| E220 | 1000000 | 35048.98 | 0.0665 | 44.81881 | 1000000 |
| E221 | 1000000 | 34838.12 | 0.0545 | 45.97761 | 1000000 |
| E222 | 1000000 | 36186.2 | 0.0505 | 44.23507 | 1000000 |
| E225 | 1000000 | 35900.27 | 0.0505 | 45.04197 | 1000000 |
| E228 | 1000000 | 34036.84 | 0.0585 | 44.94639 | 1000000 |
| E230 | 1000000 | 35666.38 | 0.0465 | 47.51582 | 1000000 |
| E234 | 1000000 | 37046.29 | 0.0505 | 44.87685 | 1000000 |
| E235 | 1000000 | 35600.72 | 0.0545 | 43.38398 | 1000000 |
| E238 | 1000000 | 37767.26 | 0.0585 | 44.13783 | 1000000 |
| E241 | 1000000 | 37046.29 | 0.0505 | 42.02605 | 1000000 |
| E243 | 1000000 | 35616.04 | 0.0585 | 44.41346 | 1000000 |
| E244 | 1000000 | 35666.38 | 0.0465 | 43.58228 | 1000000 |
| E245 | 1000000 | 37091.83 | 0.0585 | 44.90879 | 1000000 |
| E247 | 1000000 | 37188.14 | 0.0625 | 40.40653 | 411582 |
| E250 | 1000000 | 36353.93 | 0.0585 | 41.29686 | 1000000 |
| E251 | 1000000 | 35048.98 | 0.0665 | 38.87079 | 175544 |
| E254 | 1000000 | 36353.93 | 0.0585 | 45.80701 | 1000000 |
| E255 | 1000000 | 34036.84 | 0.0585 | 44.99602 | 1000000 |
| E256 | 1000000 | 37046.29 | 0.0505 | 40.40017 | 393772 |
| E258 | 1000000 | 36228.78 | 0.0465 | 46.81892 | 1000000 |
| E261 | 1000000 | 34358.92 | 0.0545 | 37.45953 | 472748 |
| E266 | 1000000 | 37767.26 | 0.0585 | 43.42947 | 1000000 |
| E267 | 1000000 | 35666.38 | 0.0465 | 45.75084 | 1000000 |
| E270 | 1000000 | 36265.76 | 0.0545 | 41.59535 | 1000000 |
| E274 | 1000000 | 31826.54 | 0.0505 | 43.99138 | 1000000 |
| E275 | 1000000 | 35600.72 | 0.0545 | 42.58948 | 1000000 |
| E285 | 1000000 | 37046.29 | 0.0505 | 42.60966 | 1000000 |
| E289 | 1000000 | 37046.29 | 0.0505 | 39.6056 | 204987 |
| E290 | 1000000 | 33345.97 | 0.0585 | 40.72702 | 608844 |
| E294 | 1000000 | 37767.26 | 0.0585 | 42.00336 | 1000000 |
| E301 | 1000000 | 35310.05 | 0.0465 | 40.73351 | 1000000 |
| E303 | 1000000 | 35682.16 | 0.0665 | 40.01482 | 382175 |
| E305 | 1000000 | 34800 | 0.04 | 50.22381 | 1000000 |
| E311 | 1000000 | 34761.92 | 0.0465 | 39.69616 | 270282 |
| E316 | 1000000 | 37767.26 | 0.0585 | 39.42296 | 354996 |
| E317 | 1000000 | 37091.83 | 0.0585 | 37.54443 | 455695 |
| E318 | 1000000 | 35666.38 | 0.0465 | 36.68229 | 367457 |
| E328 | 1000000 | 35346.64 | 0.0625 | 39.36474 | 171865 |
| E329 | 1000000 | 36015.78 | 0.0625 | 44.08938 | 1000000 |
| E273 | 987034 | 35203.93 | 0.0465 | 41.08197 | 987034 |
| E252 | 969027 | 34588.68 | 0.0465 | 42.73192 | 969027 |
| E306 | 917181 | 32840.84 | 0.0625 | 40.57078 | 296239 |
| E310 | 915238 | 30159.83 | 0.0465 | 38.52841 | 119005 |
| E338 | 908977 | 33044.89 | 0.0585 | 38.89756 | 130651 |
| E315 | 904459 | 33116.98 | 0.0665 | 36.89333 | 246452 |
| E331 | 889350 | 28780.87 | 0.0825 | 39.10757 | 196191 |
| E321 | 826061 | 29892.01 | 0.0505 | 38.46397 | 195222 |
| E248 | 786498 | 27374.43 | 0.0825 | 41.37163 | 786498 |
| E282 | 763520 | 28285.59 | 0.0505 | 39.59285 | 292628 |
| E213 | 748432 | 27404.01 | 0.0665 | 43.00253 | 748432 |
| E296 | 728516 | 25352.36 | 0.04 | 41.70019 | 728516 |
| E330 | 675619 | 25176.92 | 0.0665 | 36.30986 | 218381 |
| E284 | 660662 | 22991.04 | 0.04 | 36.71327 | 180709 |
| E257 | 647046 | 24112.15 | 0.0665 | 41.73469 | 647046 |
| E191 | 630655 | 22847.86 | 0.0465 | -52.9993 | 337858 |
| E237 | 616736 | 21928.1 | 0.0785 | 39.52755 | 105450 |
| E253 | 600993 | 19466.9 | 0.0705 | 40.23277 | 254229 |
| E276 | 582776 | 20756.17 | 0.0585 | 43.32132 | 582776 |
| E343 | 504569 | 18343.07 | 0.0585 | 32.29324 | 166705 |
| E174 | 473237 | 17553.23 | 0.0585 | 43.81291 | 473237 |
| E262 | 460433 | 16673.18 | 0.0825 | 41.44821 | 460433 |
| E300 | 438968 | 15271.29 | 0.0465 | 41.32359 | 438968 |
| E260 | 425848 | 15204.24 | 0.0825 | 38.15374 | 151438 |
| E298 | 422682 | 13841.15 | 0.0745 | 39.96768 | 234547 |
| E263 | 421317 | 14897.59 | 0.0705 | 42.57171 | 421317 |
| E271 | 400563 | 14380.32 | 0.0505 | -42.0681 | 249674 |
| E361 | 373165 | 12296.9 | 0.0465 | 35.68035 | 228707 |
| E160 | 362180 | 13678.55 | 0.0585 | 49.9538 | 362180 |
| E323 | 351215 | 13027.21 | 0.0585 | 37.41996 | 216149 |
| E407 | 350377 | 13056.79 | 0.0665 | 34.57356 | 180051 |
| E309 | 342319 | 12289.34 | 0.0505 | 39.36981 | 142970 |
| E363 | 290734 | 10761.79 | 0.0585 | 31.96846 | 138680 |
| E327 | 283139 | 10103.01 | 0.0665 | 39.66022 | 139145 |
| E357 | 283112 | 10010.72 | 0.0705 | 32.72664 | 178867 |
| E319 | 265334 | 9233.623 | 0.04 | 38.43883 | 179968 |
| E380 | 260170 | 8726.644 | 0.0625 | 39.08411 | 101394 |
| E341 | 250940 | 8701.969 | 0.0625 | 37.88419 | 112916 |
| E394 | 248308 | 8638.405 | 0.0465 | 31.28374 | 134911 |
| E350 | 246725 | 8959.904 | 0.0665 | 32.56038 | 150761 |
| E349 | 243269 | 8448.173 | 0.0585 | 44.19478 | 243269 |
| E277 | 239773 | 7901.238 | 0.0465 | 39.01249 | 122997 |
| E358 | 237240 | 8624.607 | 0.0585 | 33.24854 | 146175 |
| E356 | 228224 | 7933.504 | 0.0465 | 31.73171 | 131789 |
| E265 | 216894 | 8044.996 | 0.0585 | -37.2865 | 149193 |
| E332 | 215170 | 7217.25 | 0.0625 | 44.88285 | 215170 |
| E342 | 214229 | 7629.988 | 0.0585 | 34.83099 | 151426 |
| E370 | 210074 | 7776.084 | 0.0585 | 33.74272 | 107744 |
| E297 | 192320 | 6693.788 | 0.0825 | -46.2728 | 116341 |
| E365 | 189961 | 6596.909 | 0.0585 | 35.68649 | 106026 |
| E348 | 179743 | 6435.929 | 0.0625 | 30.63271 | 122488 |
| E351 | 175826 | 6332.511 | 0.0625 | 31.39332 | 136560 |
| E326 | 170806 | 6032.141 | 0.0665 | 35.98202 | 128634 |
| E295 | 170574 | 6194.454 | 0.0665 | 33.86126 | 135111 |
| E372 | 168139 | 5405.823 | 0.0545 | 30.45315 | 111110 |
| E308 | 167319 | 5741.839 | 0.0825 | 38.73663 | 105052 |
| E376 | 156187 | 5781.407 | 0.0585 | 31.43069 | 102989 |
| E335 | 154448 | 5755.501 | 0.0665 | 39.48853 | 124855 |
| E371 | 153438 | 5508.465 | 0.0505 | 31.44249 | 101547 |
| E180 | 153059 | 5209.644 | 0.0585 | 39.67926 | 125969 |
| E339 | 139921 | 4610.816 | 0.0465 | 34.57577 | 100797 |
| E375 | 138129 | 4843.739 | 0.0505 | 29.42261 | 101570 |
| E362 | 136749 | 4971.364 | 0.0585 | 36.86082 | 110257 |
| E406 | 135525 | 4588.532 | 0.0465 | 29.67515 | 108185 |
| E392 | 135374 | 4749.564 | 0.0505 | 30.55307 | 100620 |
| E281 | 133787 | 4553.686 | 0.0585 | -39.8663 | 101994 |
| E322 | 126297 | 4609.488 | 0.0625 | -35.7533 | 106287 |
| E391 | 117168 | 3967.01 | 0.0465 | 29.18614 | 106195 |
| E354 | 115869 | 4135.856 | 0.0465 | 32.95881 | 104191 |
| E347 | 115137 | 4133.449 | 0.0505 | 31.44108 | 103544 |
| E386 | 113109 | 3968.402 | 0.0505 | 38.32765 | 104123 |
| E364 | 110026 | 4076.055 | 0.0505 | 32.35393 | 102570 |
| E387 | 109907 | 3878.696 | 0.0705 | 27.7247 | 104616 |
| E388 | 96702 | 3368.916 | 0.0545 | 28.71603 | 96702 |
| E374 | 93243 | 3381.42 | 0.0585 | 33.32283 | 93243 |
| E368 | 87625 | 3046.013 | 0.0465 | 31.23765 | 87625 |
| E413 | 80457 | 2921.824 | 0.0665 | 28.11319 | 80457 |
| E334 | 80367 | 2836.208 | 0.0705 | 36.84138 | 80367 |
| E390 | 76640 | 2664.153 | 0.0465 | 43.0375 | 76640 |
| E355 | 72422 | 2536.984 | 0.0785 | 33.73967 | 72422 |
| E385 | 69733 | 2535.069 | 0.0585 | 32.69442 | 69733 |
| E405 | 63313 | 2259.145 | 0.0665 | 37.44804 | 63313 |
| E399 | 61465 | 2092.074 | 0.0585 | 29.74574 | 61465 |
| E398 | 58982 | 2084.815 | 0.0625 | 29.86774 | 58982 |
| E403 | 50916 | 1733.02 | 0.0585 | 31.38337 | 50916 |
| E366 | 47318 | 1646.666 | 0.04 | 30.63772 | 47318 |
| E408 | 43794 | 1592.084 | 0.0585 | 29.90416 | 43794 |
| E395 | 41454 | 1565.604 | 0.0585 | 33.35176 | 41454 |
| E410 | 40438 | 1463.298 | 0.0505 | 27.31814 | 40438 |
| E409 | 26751 | 970.1453 | 0.0545 | 25.05225 | 26751 |
| E400 | 25391 | 868.3095 | 0.0705 | 39.73471 | 25391 |
| E401 | 22356 | 771.8628 | 0.0745 | 27.98629 | 22356 |
| E344 | 20959 | 776.4533 | 0.0505 | 33.30649 | 20959 |
| E418 | 16755 | 611.5108 | 0.0625 | 29.05697 | 16755 |
| E416 | 10934 | 412.9473 | 0.0585 | 26.49884 | 10934 |
| E422 | 7617 | 269.2353 | 0.0625 | 19.20558 | 7617 |
| E393 | 5937 | 200.4331 | 0.04 | 27.62631 | 5937 |
| E417 | 5933 | 219.6155 | 0.0585 | 21.75169 | 5933 |
| E402 | 5771 | 198.6136 | 0.0665 | 35.19193 | 5771 |
| E412 | 2720 | 98.6397 | 0.0585 | 24.51094 | 2720 |

## 问题三Word2Vec模型代码

|  |
| --- |
| Word2Vec模型训练和使用 word2vec\_compute.py |
| import gensim  import logging  import numpy as np  model = gensim.models.Word2Vec.load("C:/Users/guozn/Desktop/wiki\_model")  catag = {}  catag["商业"] = ["商", "贸易", "销售", "经营"]  catag["服务业"] = ["物流", "房地产", "酒店", "服务", "事务", "招标"]  catag["工程"] = ["路", "桥", "建", "环境", "装", "林"]  catag["医药"] = ["医", "药", "卫生"]  catag["技术"] = ["科技", "电子", "通信", "网络"]  catag["文娱"] = ["文", "娱乐", "影视", "广告", "演艺"]  def makeFeatureVec(words, model, num\_features):  # Function to average all of the word vectors in a given  # paragraph  #  # Pre-initialize an empty numpy array (for speed)  featureVec = np.zeros((num\_features,), dtype="float32")  #  nwords = 0.0  #  # Index2word is a list that contains the names of the words in  # the model's vocabulary. Convert it to a set, for speed  index2word\_set = set(model.wv.index2word)  #  # Loop over each word in the review and, if it is in the model's  # vocaublary, add its feature vector to the total  for word in words:  if word in index2word\_set:  nwords = nwords + 1.0  featureVec = np.add(featureVec, model[word])  #  # Divide the result by the number of words to get the average  featureVec = np.divide(featureVec, nwords)  return featureVec  def getAvgFeatureVecs(reviews, model, num\_features):  counter = 0  reviewFeatureVecs = np.zeros((len(reviews), num\_features), dtype="float32")  for review in reviews:  if counter % 1000 == 0:  print("Review %d of %d" % (counter, len(reviews)))  reviewFeatureVecs[counter] = makeFeatureVec(review, model, num\_features)  counter = counter + 1  return reviewFeatureVecs  names = [u"商业", u"服务业", u"工程", u"医药", u"技术", u"文娱"]  center = []  for key, value in catag.items():  for i in value:  center.append(model[i])  from sklearn.decomposition import PCA  from matplotlib import pyplot  from pandas import DataFrame  # 基于2d PCA拟合数据  pca = PCA(n\_components=2)  X = center  print(X)  result = pca.fit\_transform(X)  print(result, names)  # 可视化展示  pyplot.scatter(result[:, 0], result[:, 1])  words = list(X)  for i, word in enumerate(words):  pyplot.annotate(names[i], xy=(result[i, 0], result[i, 1]))  pyplot.show() |