大学生创新创业训练计划项目

结项报告

项目名称：P2P 网贷中信贷用户违约预测模型的构建
——基于 stacking 算法的多模型融合

项目类别：大学生创新训练项目

项目级别：国家级

立项年度：2018 年

依托单位：统计与数学学院

主 持 人：郝建锋

项目成员：李小高、王燕坪、施雾珂、陈步

指导教师：潘蕊

2019 年 4 月 6 日

内容摘要：本文的研究主题为构建 P2P 网贷中信贷用户违约预测模型。随着监管的加强以及竞争的加剧，P2P 网贷作为风险频发的金融行业中的一员，经营中的高风险属性开始暴露。为了有效控制 P2P 网贷的信用风险，本文借鉴传统信贷进行风控的指标体系和一些学者的研究，搭建了一套适合 P2P 网贷借款人信用风险预测的指标体系，并依此开展数据收集和模型构建工作。综合考量下，本文选择了信用管控体系较为严格、透明的人人贷平台作为研究对象。在数据获取方面，通过 python 软件使用爬虫技术，本文获取了人人贷 2015 年 1 月 1 日到 2016 年 12 月 31 日的全部借款人数据。在数据描述方面，本文对这些数据进行了整理和分析，并利用统计图表进行了探索性分析，初步了解各自变量与是否违约的关系。在模型构建方面，为更好的进行模型拟合，本文首先进行了变量的预处理工作，主要包括对缺失值进行 k 近邻插补处理，对非平衡样本采取 SMOTE 采样处理等方面。其次，为克服交叉学习现象，本文构建模型时均采用 10 折交叉验证的网格搜索法来确定重要参数的取值。最终本文选择以 KNN 算法、随机森林、Xgboost 算法作为初级学习器，以 logistic 回归作为次级学习器，构建了基于 Stacking 集成策略的预测模型来预测 P2P 网贷平台中借款人的违约风险。在模型评估方面，本文通过 AUC 和混淆矩阵等评估方法对模型性能进行评估。经实证研究结果表明，相较单一的 KNN、随机森林、Xgboost 模型，基于 Stacking 集成策略的预测模型的预测正确率有所提升，并且在综合评价指标 AUC 评测下略强于其他模型。最后，在此基础上，课题小组结合探索性分析内容和模型的变量重要性对 P2P 网贷平台的借款人审核提出可行性建议。

关键词：P2P 网贷　信用风险　非平衡数据　stacking 算法　融合模型

Abstract： The subject of this paper is to construct a prediction model of credit user default in pear-to-pear online lending. With the strengthening of the supervision and the intensification of the competition, P2P has become a member of high-risk financial industry. For the purpose of the control the credit risk of P2P effectively, this paper draws on the traditional credit risk control index system and some scholars' research, and builds a set of index system which is suitable for P2P online borrower credit risk prediction, and collects data as well as builds model. After the comprehensive consideration, this paper chooses Renren loan platform as the research object as its credit management system is strict. In terms of data description, we analyze these data and use the statistic charts to conduct the exploratory analysis to get a preliminary understanding of the relationship between the independent variables and whether they are defaulted. In terms of model construction, we firstly carry out the data preprocessing, including K-nearest neighbor interpolation for the missing values. Secondly, in order to avoid the cross-learning phenomenon, we use a 10-fold cross-validation grid search method to determine the value of the parameter. In the end, we choose the KNN algorithm, random forest and Xgboost algorithm as the primary learner, the logistic regression as the secondary learner, and construct a predictive model which based on the stacking integrated model to predict the default risk of the borrowers in P2P online lending platform. In terms of model evaluation, we use AUC and confusing matrix to evaluate the performance of the model. The results show that the stacking integration model is slightly stronger than other models. Finally, on this basis, we combined the exploratory analysis and the stacking model to propose some feasible suggestions for the P2P online lending platform.

Key Words： pear-to-pear online lending credit risk imbalanced data stacking algorithm fusion model

目　　录

引言

作为一类互联网金融借贷业务平台，因契合小贷市场的需求，P2P 网贷平台近些年处于快速成长期。然而，随着监管的加强以及竞争的加剧，作为风险管控极为困难的行业之一，其经营中的高风险属性开始显现。

目前我国 P2P 网贷的信用评价体系仍不完善，而且对 P2P 平台和信息的监管力度不够，平台公司鱼龙混杂，规模参差不齐，有恶意放贷、融资的可能，个人基本信息也有非法买卖或被盗用的风险，对金融市场的稳定产生不利影响。我国信用评价体系较其他发达国家起步晚，目前央行主导的征信体系还无法适用于 P2P 网贷平台，也不可能跟得上互联网金融的发展速度。因此，为了有效控制 P2P 网贷的信用风险，本文借鉴传统信贷进行风控的手段和一些学者的研究进行了以下研究：

1. 探究 P2P 网贷现状

课题小组通过观察多个 P2P 网贷平台运营情况，发现各平台都不同程度的存在借款人违约行为，而过多的违约现象可能会导致平台出现资金链断裂的情况，影响互联网金融的健康发展。因此，减少网贷违约行为变得极为重要。为了解决上述问题，课题小组从人人贷平台上获取 2008 至 2018 年的相关借贷数据，包括借款人的年龄、职业、借款金额、借款利率、历史借款总额等多方面信息。分析我国 P2P 网贷平台的总体状况以及所获取的数据总结出其违约率较高的原因。

2. 完善 P2P 网贷领域指标体系

传统金融机构进行信用评估时侧重于对借款人还款能力的评估和信贷记录的考核，然而这种评估时效性差，且中国尚未建立全民的信贷记录数据库，采集有一定困难，并且从小组实际采集的数据来看，该类指标覆盖率较低，因此传统的指标体系不适合完全应用于 P2P 网贷领域。分析发现借款信息和历史信用信息都是 P2P 平台自身运营过程中用户行为所产生的数据，这些信息与平台运营高度相关却未被纳入传统体系考虑范围。因此，在传统金融机构信用评估体系的框架上，广泛参考专家研究成果，从个人基础信息、借款信息、资产信息、历史信用信息四个方面刻画借款人属性。

3. 构建违约预测模型

目前违约预测方面多使用分类模型算法，主要包括逻辑回归方法、决策树法等统计模型，也包括神经网络等非统计模型。但单个模型可能产生的欠拟合和过拟合问题，不能兼顾高预测精度和稳健性，为更好地平衡模型的预测精度和泛化能力，课题小组选择以 KNN 算法、随机森林、Xgboost 算法作为第一层分类器，以逻辑回归作为第二层分类器，创建了基于 Stacking 算法的违约预测模型。

4. 实证分析与建议

通过混淆矩阵分别比较所构建的单一模型和集成模型的查全率、查准率和 F1 等指标，来反映课题小组构建的模型分类预测在实际应用中的表现，并通过模型输出结果分析变量重要性，重点研究对模型类别划分结果影响较大的变量，结合探索性分析探索这些变量对违约与否的影响程度和方向。

一、文献综述

（一）风险控制理论

自 P2P 平台逐步兴起以来，就有许多学者开始探究关于 P2P 网贷平台借款人特征对借款成功几率以及其违约率的影响。

2004 年，国外学者 Peterson 首先提出，对借款方的审核应包含收入、职业、资产、人际关系等硬性及软性指标。2008 年，Klafft 发现，在借款人的各种经济状况指标中，收入对获取资金有清晰的影响。与此同时，Herzenstein 等人经过论证得出了借款人的生活情况和家庭背景对降低违约均是积极作用。接着 Lin（2009）发现通过人际关系和社会成本可以对借贷风险进行有效控制。

2012 年起，国内也陆续开展了相关探究。王梦佳（2014）运用回归分析的方式证实借款人的学历对其还款率有着正面影响，而年龄和职位的影响作用不大；在家庭方面，婚姻生活越正常，违约概率越低。王会娟，廖理（2014）论证后得出了借款人的信用评级越高，就越有可能借到钱，违约率也会降低。李云霞等人（2016）基于真实状况进行实践建模分析后，也得出了同样的结论。

综合来说，虽然网贷的崛起没有多久，但引发了诸多人士的关注，人们进行了大量的研究，帮助 P2P 行业不断进步。从这些报告中可以看出，关于网贷平台的探讨大多数都停留在对借款人的个人信用方面的评估，一方面是对个人信息的评估，例如性别、居住城市等；另一方面是对资产特征的描述，例如是否有房贷、车贷等。

（二）非平衡数据理论

类别不平衡即在机器学习分类工作中，不同种类的数据在数量上有显著的的区分。对类别不平衡数据使用传统学习机分类，往往不能有效地少数类数据的蕴含信息，结果分类精度不高、查准率、查全率、AUC 值等指标较差，并且数据量较大的一类会有比较小的错分代价。Japkowicz 与 Stephen 于 2002 年提出类别不平衡对传统学习机的负面影响因素可概括为由类别不平衡比率（IR）、重叠区域大小、训练样本的绝对数量、类内子聚集想想的严重程度、样本噪声比率以及样本维度等方面。以朴素贝叶斯分类器为例，类别不平衡样本的 IR、重叠区域越大，其决策结果的分类也将越偏离实际结果。本研究旨在预测 P2P 网贷借款者是否会违约，将借款者分为不会违约和会违约两类。从实际收集的数据来看，违约样本的样本量相对较小，出现类别不平衡现象。因此直接对样本数据使用传统学习机进行分类得到的结果并不令人满意。

不平衡数据问题可以采取以下两种方式解决，这两种方式分别有不同的侧重点，一类方法侧重数据，因为数据是非平衡的，可以采取合适的抽样方法，以减小样本的不均衡性；另一类方法侧重算法，通过赋予不同错误分类情况不同的代价，使得算法能够在不平衡的样本训练下也可以输出较为满意的结果。

其中从数据角度处理不平衡样本的方法又可以细分为以下几种：（1）随机下采样，从多数类中随机选取一定量的数据与少数类组合在一起，构成新的样本。这种措施容易操作却有弊端，在随机抽取的过程中，容易遗失数据中的相关信息，尤其是数据总量不太大的情况下，样本信息丢失更为严重。（2）随机过采样，从少数类中随机选取一定类的样本进行复制与多数类组合在一起，构成新的样本。同样，这种方法操作简便，而且新生成的样本数量也较多，但是其也有不小的缺点，首先其完全复制少数类样本，容易造成少数类数据分布的偏误，使得少数类数据的分布逼近初始少数类样本点的离散分布；其次在复制过程中，其难免也会复

制其中的噪声，使得噪声信息被放大。（3）SMOTE，这是一种以过采样措施为基础的改良办法，其主要工作是对种类较少的数据进行学习进而依照学习结果进行人工合成新数据，这种方法能够有效减轻过采样附加的过拟合现象。

从算法角度解决数据不平衡的措施主要为使用代价敏感的学习办法，代价敏感学习可以从构造代价敏感学习模型、基于分类结果的后处理和改变原始样本的初始分布等方面减小不平衡样本的影响，提高模型的效果。

（三）数据挖掘理论

刘云焘等（2005）将支持向量机运用到银行的风控评估中，实证研究中发现相对于 BP 神经网络，支持向量机具有规则易行、更加稳健、准确度高、鲁棒性高等特点。Chitra 和 Subashini（2013）将支持向量机（SVM）、决策树（DT）、logistic 回归方法运用到商业银行信用风险领域，但研究并未指出最优模型。Kambal 等（2013）将决策树（DT）和人工神经网络（ANN）用于信用评价，发现 ANN 优于 DT，但 DT 的结果解释性更强。

吴冲等（2004）研究创立以模糊神经网络为学习器的商业银行风控测评模型，研究证明模糊神经网络预测误差更小，适合用于商业银行风控测评。Bhattacharyya 等（2011）分别采用支持向量机（SVM）和随机森林（RF）预测信用卡欺诈，研究发现 RF 模型表现较好，但 SVM 模型效果不如逻辑回归，但他们认为这可能是由变量选择不合适造成的。

由于单一模型在预测过程中有各自的局限性，集成算法被引入解决信用预测问题。

丁岚等（2017）将逻辑回归算法、DT、SVM 选为初级模型，并将 SVM 选定为二级模型，创立了依赖于 Stacking 算法的预测模型来预测网贷中借款方是否能够及时还清贷款，研究表明较于单一模型，依赖于 Stacking 算法的预测模型可以达到明显提升模型评价中的正确率的作用。Pang 等（2014）将支持向量机与粒子群优化算法的混合模型引入个人信用等级划分。夏雨霏等（2015）研究发现聚类支持向量机在信贷领域分类表现明显变好，误判成本将校，比较适合实际应用。Huang 等（2006）等集成了遗传算法与支持向量机的混合模型并运用与个人信用等级划分，该混合模型对数据的预测精度有明显提升。

二、研究思路与建模策略

（一）模型集成策略

目前可以用于分类问题的算法有许多种，比如：支持向量机、KNN 法、Logistic 回归、决策树算法以及一些集成类算法等。从相关文献以及课题小组对单一模型的构建过程中发现，众多单一模型在分类准确率、稳健性和可解释性等方面各有偏重。如 Logit 方法易于理解和实现，而准确率偏低，且一般不能直接对不均衡样本使用；决策树方法有较好的拟合精度和可解释性，但稳健性不足。故在本次研究过程中，为了平衡稳定性与准确率将的关系选择了 Stacking 集成方法：

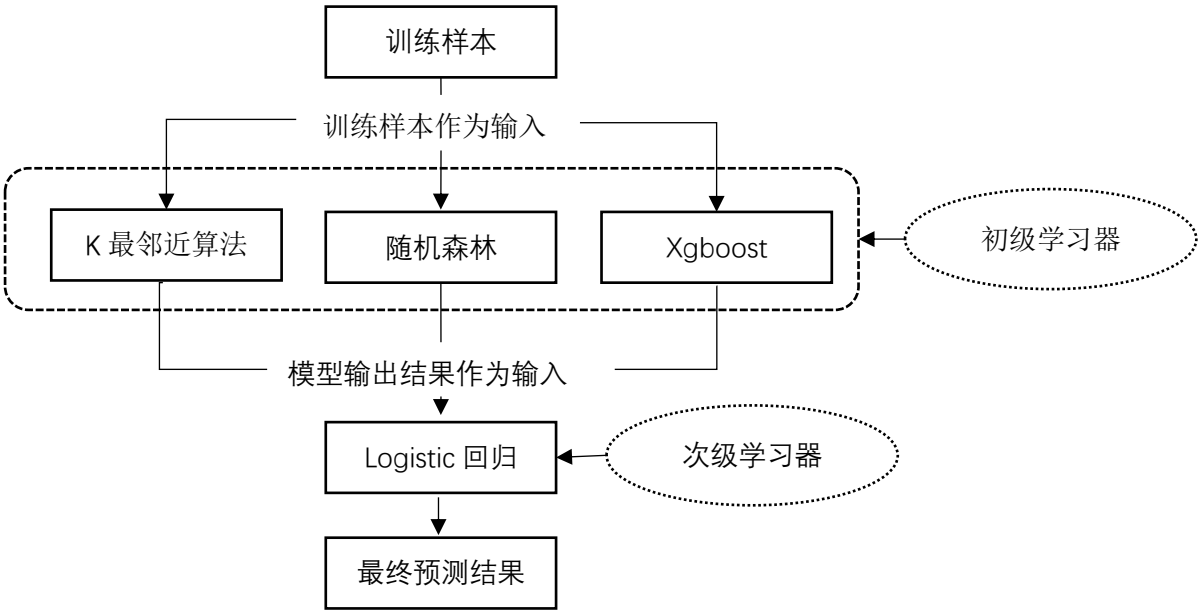

图 1 模型集成策略

在具体的研究中，首先以 K 最邻近算法、随机森林和 xgboost 作为初级学习器，以经过预处理后的训练样本作为输入来训练这三个模型，三个初级学习器的输出结果为每个样本是否违约的预测概率。接下来将这些概率作为刺激学习器 logistic 回归的原始数据，对实际类别“是否违约”进行学习以得到最终的预测模型。

（二）模型评估策略

1. 混淆矩阵衍生指标

在不失一般性的情况下，本文以二分类的情况介绍相关模型评估指标。以 1 代表正类，0 代表负类。一般以 1 代表人们更关注的类别，比如信用欺诈问题中，将发生信用欺诈行为的的类别设为 1。在进行模型评估时，混淆矩阵包含大量的信息，相对于精度而言，它更能体现出一个学习器的多方位的能力。表 1 为混淆矩阵常见形式。

表 1 混淆矩阵

Confusion matrix		Predicted condition	
		Positive	Negative
True condition	Positive	TP	FN
	Negative	FP	FN

混淆矩阵常见的指标计算：

- Precision=TP/(TP+FP)，代表查准率。
- Recall=TP/(TP+FN)代表查全率。

一般情况下，上述两种指标时常呈现相反的变化。这种现象可以这样解读，为了提高查准率，学习器倾向于把更多的样本归为负类，导致 FP 减小，FN 增大；若是为了提高查全率，学习器倾向于把更多的样本归为正类，导致 TP 增大，FP 也增大，但 FP 得增量将会比 TP 得增量大许多，所以导致查准率降低。

为了平衡两个指标在模型评估过程中的作用，F1 度量指标被引入，计算公式如下：

$$F_1 = \frac{2 \times P \times R}{P + R}$$

但有时对于不同的分类问题，其关注的指标也略有差别。比如，利用机器学习判断蘑菇是否有毒时，查全率更为重要，因为毒蘑菇被错分为无毒蘑菇的后果更为严重，将会导致食用者中毒。

因此 F_β 被引入，其计算公式如下：

$$F_\beta = \frac{(1+\beta^2) \times P \times R}{\beta^2 \times P + R}$$

其中 β ＞ 0,当 β ＞ 1 时，recall 的影响较为明显；当 β ＜ 1 时，precison 的影响更加明显。

2. RCO 曲线与 AUC 值

首先看一下一些学习器的特点，许多学习器能够基于样本产生一个在 0-1 间的小数，这个小数可以代表样本划分为类别 1 的概率，可以将这些实值进行从大到小排好，然后设置一个阈值，比它大的样本类别归为类别 1，比它小的样本类别归为类别 0。从这里可以看出，学习器产生的一系列的实值对学习器预测效果的好坏至关重要，一个好的学习器应该使得实际类别为 1 的样本对应的实值尽可能大，以确保其被分类为 1。

ROC 曲线的绘制：首先根据前面所说的一些学习器能够能够产生一系列类似于概率的实值计算横轴指标 FPR=FP/(TN+FP)，纵轴指标 TPR= TP/(TP+FN)，然后根据设定阈值的不同在作图区域找到对应的点，并连接起来。

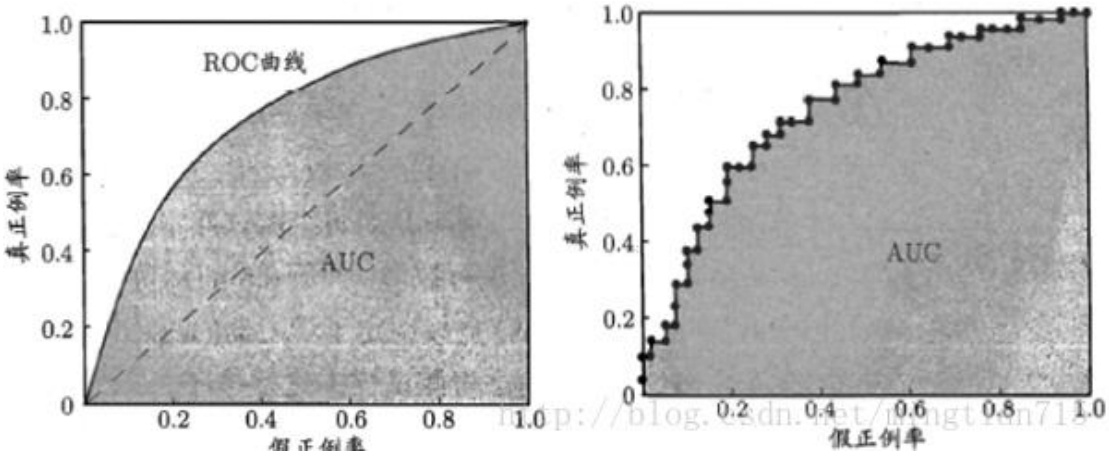

图 2 ROC 曲线与 AUC

图 2 中左图为理想的 ROC 曲线，因为在现实中，样本的数量经常较少，右图是更常见的情形。

ROC 曲线分析：对角线属于纯靠随机猜得到的结果，样本划分正确与错误的概率相同，所以一个训练好的模型的 ROC 曲线必须包住对角线才值得使用。在比较模型评价时，假定其中一个模型的 ROC 曲线可以遮住其余模型的 ROC 曲线，则认为这个模型比较好。

AUC：ROC 曲线与两个坐标轴所围成的封闭图形的面积。通常来讲，AUC 的值较高，说明学习器对数据的分类排序能力较好，所以对于 ROC 曲线有重叠交叉现象发生的分类器，便可以通过 AUC 进行比较，AUC 值较高的学习器相对表现较好。

三、探索性分析

（一）数据基本说明

1. 数据来源

虽然国内拥有较多的 P2P 网贷平台，但是许多平台对于借款人的交易数据披露过少，这给研究带来了不小的困难。通过一番比较，发现人人贷平台的数据量较大而且信息较为齐全，因此本次研究选取了人人贷平台的借贷数据作为数据来源。

借款人是否违约受多种因素影响，本研究利用网络爬虫从人人贷平台获取了相关的数据，经整理编制变量说明表如表 2。

表 2 变量说明表

		变量名	取值范围
因变量		标的状态	逾期、还清
自变量	标的信息	标题	文本数据
		标的总额	3000-48000 元
		年利率	7.0%-13.2%
		期限	2015.01-2016.12
	信用信息	申请借款	1-42 笔
		成功借款	1-40 笔
		还清笔数	0-40 笔
		借款总额	300-1960000 元
		逾期次数	0-38 次
	基本信息	年龄	18-61 岁
		性别	男、女
		学历	高中或以下、大专、本科或以上
		婚姻	未婚、离婚、已婚
		收入	<0.5、0.5-1、1-2、2-5，>5（万元）
		公司规模	10 人以下、10-100 人、100 人以上
		工作城市	除港澳台外 31 个省
		工作时间	1 年以下、1-3 年、3-5 年、5 年以上
	资产信息	房产	无、有
		房贷	无、有
		车产	无、有
		车贷	无、有

共计 30332 条借贷数据，包含 26 个变量，本研究将通过分析这些数据来探索借款人是否违约与其他因素的关系。

2. 数据预处理

变量预处理包括标准化、重编码、数据集划分、异常值处理、非平衡数据处理等多方面内容，本文挑选几项工作来进行说明。

（1）变量分组

为了增加模型的鲁棒性，在数据预处理的过程中，对部分变量进行了分组和并等方法，在此对其进行详细说明，表 3 中括号内为借款人数。

表 3 变量分组说明表

变量名	处理前	处理后
收入	1000元以下(92) 1000-2000元（59） 2000-5000元（8201） 5000-10000元（11347） 10000-20000元（5382） 20000-50000元（3374） 50000元以上（1875）	5000元以下（8352） 5000-10000元（11347） 10000-20000元（5382） 20000-50000元（3374） 50000元以上（1875）
学历	高中或以下（5435） 大专（14610） 本科（9653） 研究生或以上（632）	高中或以下（5435） 大专（14610） 本科或以上（10285）
婚姻	孤寡（107） 离婚（2808） 未婚（6998） 已婚（20417）	离婚（2915） 未婚（6998） 已婚（20417）
公司规模	10人以下（20948） 10-100人（4664） 100-500人（1653） 500人以上（2469）	10人以下（20948） 10-100人（4664） 100人以上（4122）

收入：关于收入变量，由于 2000 元以下的两个类的取值样本数过少，因此将 1000 元以下、1000-2000 元和 2000-5000 元三个属性合并，记作 5000 元以下。

学历：关于学历变量，由于研究生或以上类的取值样本数过少，因此将研究生或以上类与本科类合并，记作本科或以上类。

婚姻：关于婚姻变量，由于孤寡类的取值样本数较少，将其与离婚类合并，记作离婚。

公司规模：关于公司规模变量，由于 100-500 人（违约率 24.2%）和 500 人以上（违约率 26.0%）两类公司规模的违约率相近，与 10 人以下（违约率 1.2%），10-100 人（违约率 16.6%）相差较大，因此将 100-500 人和 500 人以上两个类合并，记作 100 人以上。

(2) 缺失值处理

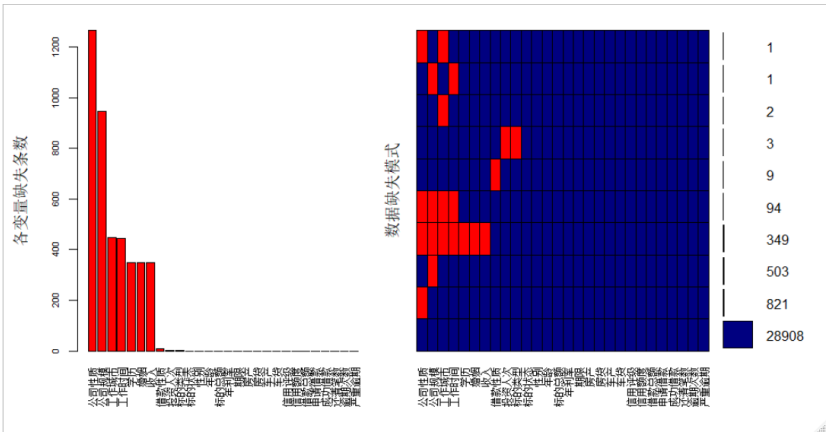

图 3 缺失值模式

2.1 删除处理

缺失值方面，从图中可看出，本次研究的数据中有 443（349+94）条数据缺失较为严重，这 443 条数据同时缺失了学历、婚姻、收入、公司规模、工作时间等变量。由于这些条数据缺失信息严重，超过变量总数的 5%，记录条数相对总的数据而言数据量较小，而且删除这些数据后对数据结构几乎没有影响，故选择将这些数据删除。

2.2 knn 填补

针对其余缺失值采取 k 值为 5 的 knn 填补法，在全体数据中找到 5 个与它最相似的对象，然后用这些相似对象的平均值来进行填充，有效保留数据的原始信息。

3. 非平衡数据的处理

为进行建模做准备，本文对非平衡数据采取改良后的 SMOTE 算法采样处理，对小类样本进行过采样，处理前的训练集样本共 23127 条记录，其中包含违约记录 1658 条，处理后共 9948 条记录，处理前违约占比 7.2%，处理后二者比值为 1。

具体执行过程如下：

输入：$S=\{(x_i, y_i),\ i=1,\ 2,\ 3,\ \cdots,\ n,\ y_i\ \in\{-1,+1\}\}$，输出样本数为 N，紧邻同类样本数为 k

输出：$S'=\{(x_i, y_i),\ i=1,\ 2,\ 3,\ \cdots,\ N,\ y_i\ \in\{-1,\ +1\}\}$

流程：

1) 设置一个空集合S_0

2) For i in 1:N-n

2.1 从少数类样本中随机选取一个样本x_i

2.2 求出x_i周围最近的 k 个少数类样本

2.3 从 1.2 中求出的 k 个样本中随机选取一个，记作x_s，计算出$x'=x_i+\text{random}[0,\ 1]\times(x_s-x_i)$。

2.4 令$S_0=S_0\cup x'$

3) 令$S'=S+S_0$，即得所求数据。

（二）数据概要生成

以下分析在引入虚拟变量、非平衡数据处理等工作前完成，主要是数据的探索性分析。

1. 基本信息

（1）违约分布分析

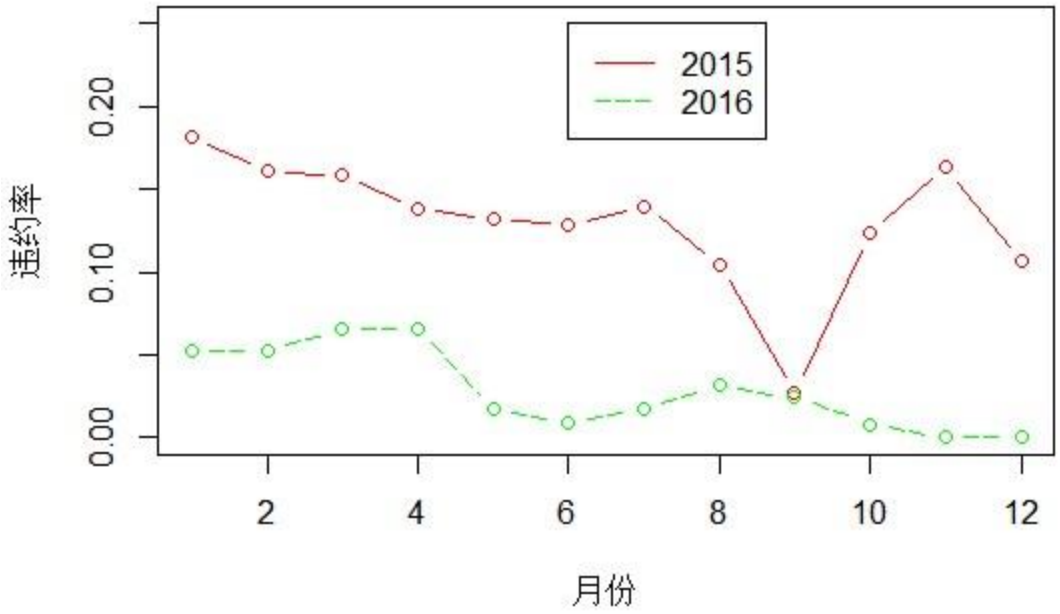

图 4 违约分布折线图

图 2 显示了 2015 年和 2016 年人人贷平台每个月的借款人中的违约率，从图中可以发现，2015 年从 7 月份至 9 月份的违约率出现较大幅度的下降，这是因为在当年 7 月份，我国第一次对网贷领域下达纲领性文件，主要涉及部门监管以及资金安全，其对网贷平台形成了较强的监管作用。

其次，2015 年和 2016 年的违约率都处于波动之中，但 2016 年相对和缓，而且其违约率相较于 2015 年，明显偏低。一方面由于数据采集的时间是在 2018 年，因此有一定比例的 2016 年的借款交易还处于还款状态中，并未纳入本次研究的范围，所以这对 2016 的违约率可能有一定的影响；相关文件陆续下发，进一步使得网络借贷市场准入、管理等有了更为明确的规范，产业内从业平台、担保机构、托管银行、第三方支付通道及信用服务平台等多方参与、互相监督的产业形态相对成型。

（2）个人信息分析

① 学历及婚姻状况与违约

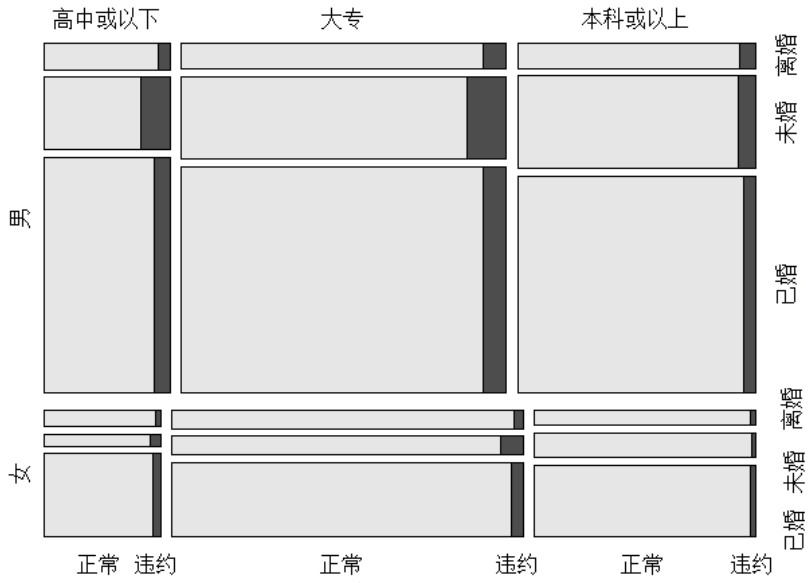

图 5 学历及婚姻状况与违约关系马赛克图

图 3 中包含了性别、学历、婚姻和是否违约等 4 个变量，从图 4 中可以发现一下信息：

- 男性借款人数明显高于女性，且男性违约率高于女性。
- 学历方面，大专学历的借款人占比较高，有较高学历的借款人违约率稍低一些。
- 婚姻方面，已婚状态的借款人占比较高，而未婚状态的借款人违约率稍高。

② 公司规模和是否违约

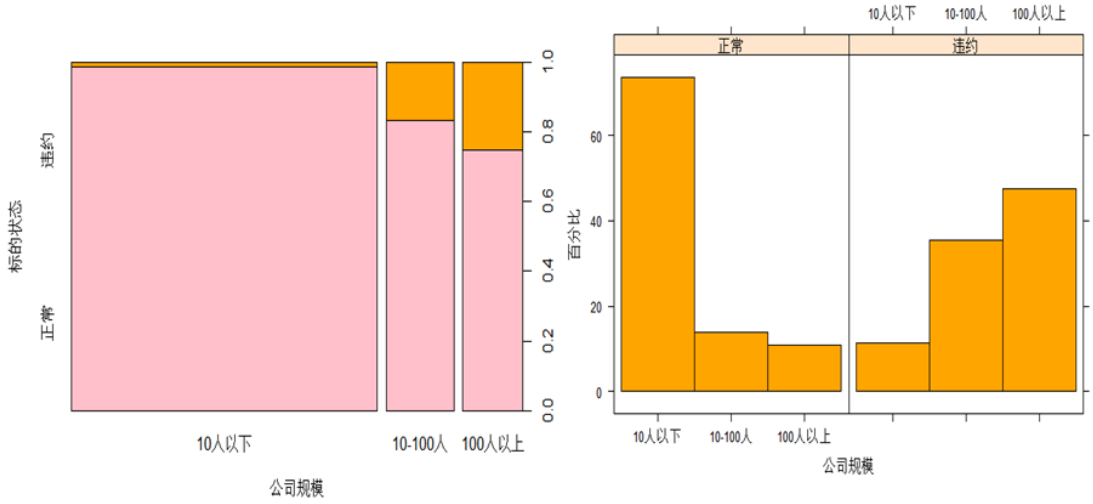

图 6 公司规模与是否违约关系图

图 4 反映了公司规模与标的状态的关系，从左边的棘状图中可以看出属于 10 人以下公

司规模的借款人占比较高，而且随着公司规模的逐渐增大，借款人的违约率逐渐升高。

从右边的柱状图中可以看出正常借款人中，10 人以下的公司规模占比较大，百人以上的公司规模占比较小；而在违约借款人中恰恰相反。

③ 年龄与是否违约

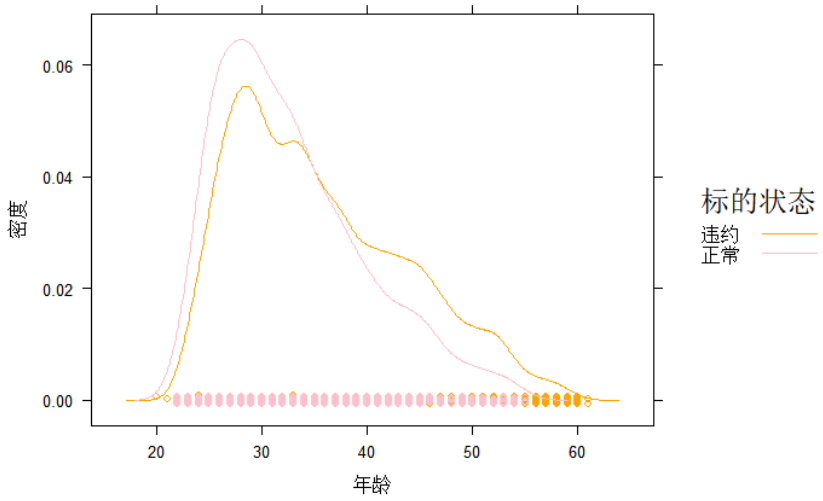

图 7 年龄对是否违约的分组密度曲线图

图 5 展示了借款人年龄与标的状态的关系，从图中可以发现，借款人的年龄主要分布在 25-40 岁之间，这类人群一般来说身体健康状况良好，工作比较稳定，有能力进行还款，因此相对来说，更容易通过网贷平台的审核，同时这类人群也更加乐意消费，对资金的需求也大。

违约借款人和正常借款人的年龄整体分布相似，但是可以明显看出，在 40 岁以上的人群在违约借款人中占比高于正常借款人，一方面 40 岁以上人群随着年龄的增长还款能力逐渐降低，导致违约；另一方面，这类人群大概由于守信意识不强，还款意愿不高。

（3）借款信息分析

① 年利率与是否违约

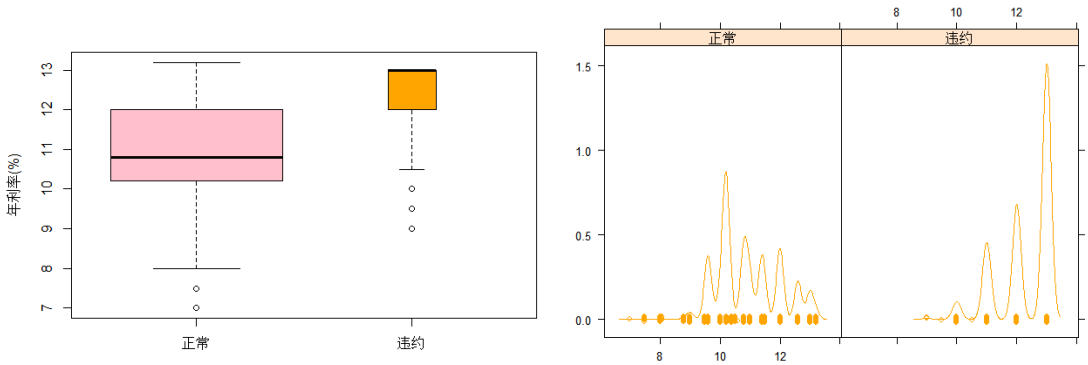

图 8 年利率对是否违约箱线图与密度图

图 6 反映了标的状态与年利率的关系，从箱线图中可以看出发生违约的交易年利率整体偏高，且分布相对集中。从年利率的密度分布图中可以看出违约交易的年利率主要分布于12%以上，而正常交易的年利率主要在 10%和 12%之间。

② 期限与是否违约

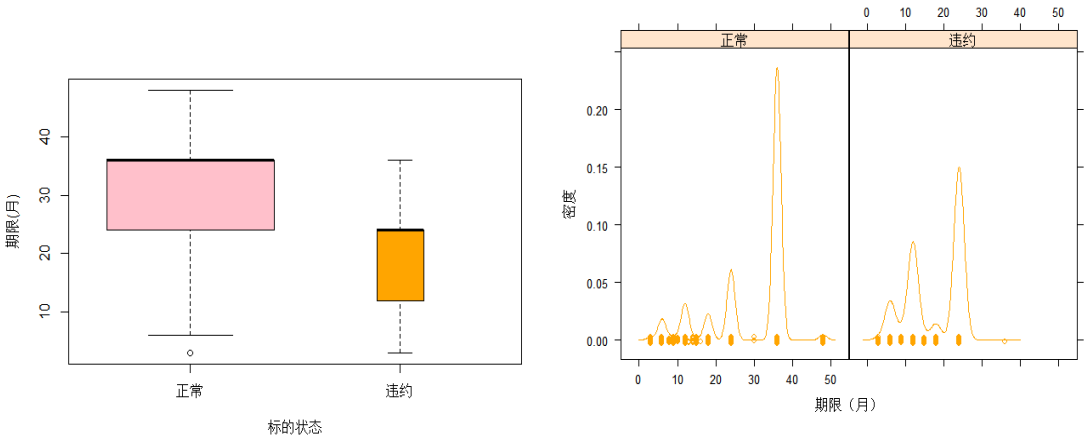

图 9 期限对是否违约箱线图与密度图

图 7 反映了标的状态与期限之间的关系，从箱线图中可以看出违约交易的中线整体偏低，而且分布较为集中。从期限的密度分布图中可以看出违约借款人的还款期限主要在 10-30 月之间，而正常借款人的还款期限主要分布在 30-40 月之间。

（4）信用信息分析

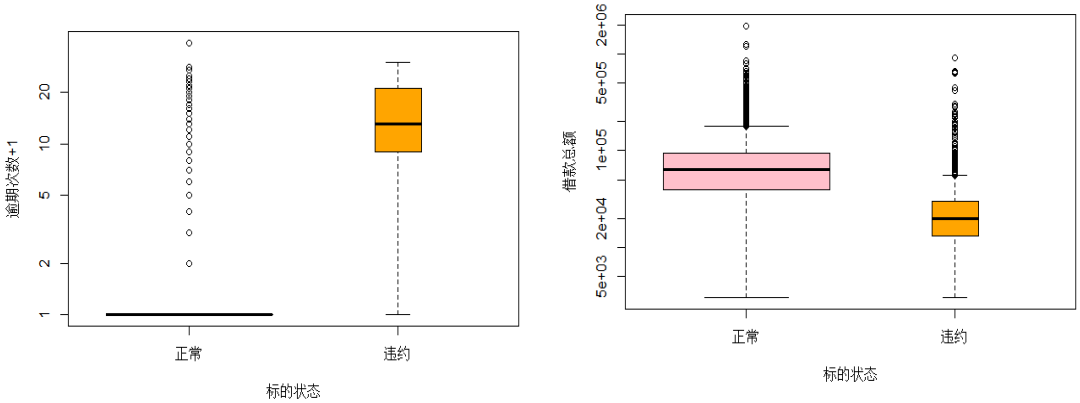

图 10 逾期次数与借款总额对是否违约的影响

图 8 反映了借款人的历史信用信息与标的状态的关系，逾期次数方面，违约借款人的逾期次数比正常借款人要多得多，逾期次数主要集中于 10-20 次之间，而正常借款人逾期次数中有较多异常值，其逾期次数主要为 0 次。借款总额方面，逾期借款人的借款总额低于正常借款人，二者都有较多的异常值。

2. 地域信息

根据借款人工作省份信息绘制成违约率的地理分布图，图中颜色深浅表示违约率的高低，颜色越深，表示违约率越高，红点表示该省每百万人中的借款人数，红点越大表示每百万人中借款人数越多。

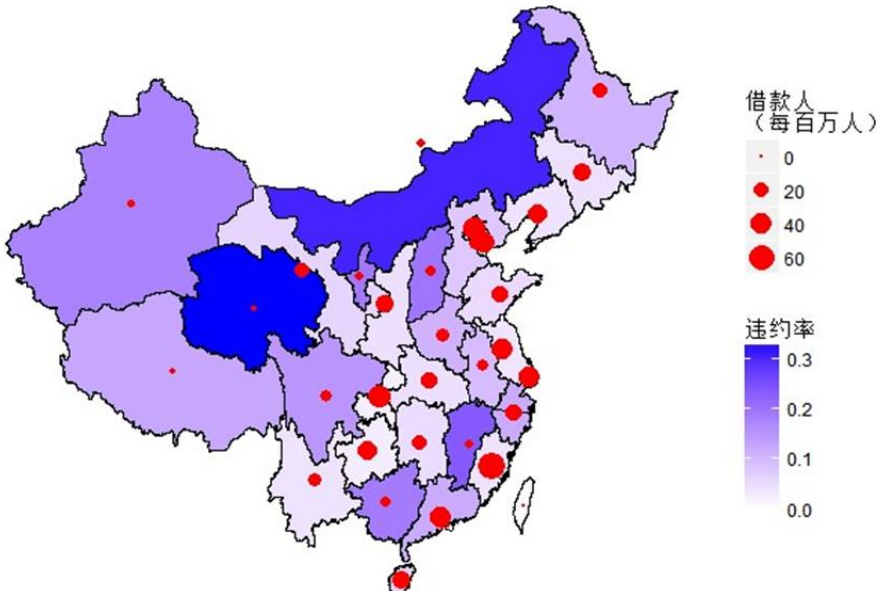

图 11 违约率的地理分布图

从图中可以发现，东部沿海省份及中部省份的借款人数较多，而西部及西北地区的省份借款人数较少。但是在违约率方面，与此相反，西部以及西北部的省份违约率较高，而东部沿海以及中部省份违约率较低。这可能与西部及西北地区的省份借款人数较少有关，由于基数较小，容易导致违约率变高。也有东部经济较为发达，人们重视信用经济的因素。

3. 文本信息

图 3 根据标题信息以词云的形式展现了违约借款人和一般借款人的借款原因。

图 12 借款原因词云图

可以看到，二者有共同的特点，在关键词中，都含有借款、消费等；二者也有不同点，违约借款人的借款原因更多的涉及房屋、个人消费、购车等用于满足个人需求的活动，而一般借款人的借款原因更多的涉及经营、生产、原材等用于创造价值的生产活动，这也在一定程度上反映了违约借款人与一般借款人的区别。

四、建模预测分析

（一）K 最近邻模型

1. k 最近邻算法简介

K 最近邻方法在确定某一样本类别划分情况时主要考虑距离其较近的 k 个样本，由这些样本的类别属性决定该样本最终的类别，这种方法简便易行。

同时为了保证初级学习器具有较大的多样性，本次研究也选取了传统方法中的 K 最邻近算法，其实现流程如下：

1) 计算测试数据与各个训练数据之间的距离
2) 按照距离的递增关系进行排序
3) 选取距离最小的 K 个点
4) 确定前 K 个点所在类别的出现频率
5) 返回前 K 个点中出现频率最高的类别作为测试数据的预测分类

2. k 最近邻模型的训练及调参

KNN 主要有 3 个参数值得考虑：

（1）k 是其中的重要参数之一，不同的 k 值代表在预测样本类别时选取的距离其最近的样本数的不同，一般这种不同在一定程度上也会影响到样本类别的判定，如上图所示，当 k 取值为 3 时，待测样本被判断为红色类别，当 k 取值为 5 时，待测样本被判断为绿色类别。所以为了保证样本在预测时尽可能高的正确性，需要选择合适的 k 值。在调参的过程中，设置 k 的调参区间为[1, 15]，其中步长为 1，经过模型对不同的 k 值对应模型的准确率的比较，最终确定 k 为 2，在这步中模型此值对应的准确率最高。

（2）权重（weights）的考虑：一种方法是将样本周围的 k 个最近的值视为拥有相同的重要性，这是 knn 中一种可选的做法；另一种方法是考虑权重问题，由于 k 个距离待测样本的最近的数据与待测数据的距离并不完全相同，因此可以将距离看做待测样本与 k 个数据之间亲密关系的一种衡量，根据距离的不同设置不同的权重，这也符合一般的直觉理解。其中权重有两个可选择的参数值，所有的样本权重相同（uniform）和与距离相关的权重（distance），通过模型准确率的对比，取与距离相关的权重（distance），此时模型的准确率更优一些。

（3）距离计算方式。既然 KNN 依靠距离作为待测样本判别的一种工具，就需要定义距离。不同的距离计算方式可能会影响待测样本的预测结果，尤其是当 KNN 考虑到权重时，不同的距离计算范式对于权重也会有一定的影响。可以选择的距离为闵可夫斯基距离，将对应 p 分别为 1,2，3,4,5,6 闵可夫斯基距离作为该参数的寻优区间，最终确定的 p 为 1 的闵可夫斯基距离。

通过一系列调整之后，KNN 算法最终确定的参数如下：其中最邻近样本数 k 为 2，选择的距离计算方式为曼哈顿距离，权重选择方式为和距离有关的权重，其中在训练样本上准

确率的最优取值为 94.98%。

（二）随机森林模型

1. 随机森林算法简介

（1）Bagging 方法：

1) 从数据集 D 中取样（放回选样），总共执行 t 次
2) 针对每一次取样训练得到分类模型，最终得到 t 个模型 $h_1, h_2, \cdots, h_t$
3) 对未知样本 X 分类时,每个模型都得出一个分类结果，得票最高的即为未知样本 X 的分类
4) 也可通过得票的平均值用于连续值的预测

（2）随机森林：随机森林在建立基学习模型时不像一般装袋法，并不会选择所有的属性，其会在属性中随机抽取 k 个属性，这个做法使得随机森林的每个基学习模型有了更大的变异性，效果也更优，其算法过程如下：

输入:数据集 $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)，\ y_i \in \lambda\}$，属性集 A，基学习算法 ϕ，训练轮数 T，每棵树选择的属性个数 k，λ 为类别集合

输出：随机森林 $H(x)$

流程：

1) for t=1, 2, ⋯, T

   1.1 从属性集 A 中随机选取 k 个属性

   1.2 利用自助法随机选取 m 个只包含相应属性的样本，记作 D_{bs}

   1.3 $h_t = \phi(D_{bs})$

   end for

2) $H(x) = \underset{y \in \lambda}{arg\ max} \sum_{t=1}^{T} I\,(h_t(x) = y)$

2. 随机森林模型的训练及调参

随机森林是 bagging 方法中一种非常有效的算法，它首先训练多个不同的决策树分别对样本进行预测，然后将这些决策树的集成，对其所包含的决策树的结果进行综合处理得到一个更准确的预测结果。

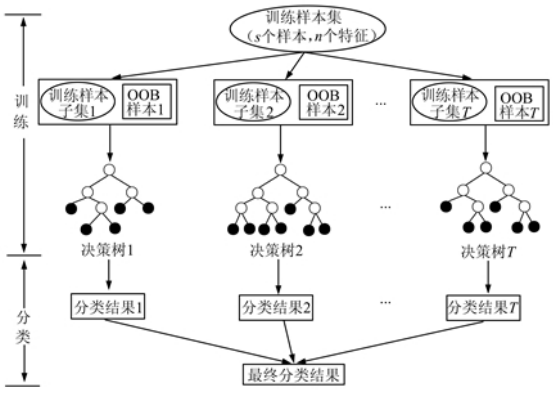

图 13 随机森林原理展示

在决策树中,有以下参数需要进行不断调整优化:首先从整体来看需要有这些参数调优：

（1）随机森林中树的个数（n_estimators），一般来说，如果选取的树的数量太少，可能或导致模型欠拟合，拟合效果较差，然而如果树的数量选取过多，一方面当树的数量很多时，随着树的数量的提升，模型的提升效果不太明显，另一方面，太多的树也会导致模型运算量增大，时间和经历成本的显著提高。在实际建模过程中，首先调节的参数就是树的个数，选择将其他变量固定为程序默认的参数，将树的个数（n_estimators）利用网格搜索方法从 50 到 100 以步长为 1 进行搜索，最终确定树的最优个数为 97，其在训练集上的测试准确率为 98.15%，可以说这个预测准确率已经很高了。

（2）内部结点再划分最小样本数（min_samples_split）和最大深度（max_depth），设置内部结点再划分最小样本数是为了防止树发生过拟合问题，通过设置这个参数，当结点的个数小于该值时，则停止树权的划分，可以防止树在产生的过程中生出一些乱七八糟的小权，导致树过于复杂。树的最大深度关乎树的结构复杂与否的问题，简单地可以理解为树的层数问题，可以从实体的树进行类比，一棵树的层数越多，则其越复杂，但是对于建模而言，树的深度越大，并不一定代表树模型越好，因为这可能已经代表树发生了过拟合现象。因此，设置树的最大深度，可以在建模的过程中，减小树发生过拟合的影响。之所以把这两个参数放在一起，是因为这两个参数从不同的方面防止了数据对训练样本的拟合过于严重的问题。因为上一步已经求出来树的最优的棵数为 97，在这步求这两个参数的过程中，依然把树的棵数（n_estimators）定为 97，其结点划分最小样本数在区间[40, 45]中选取，而最大深度在区间[10, 20]中选取，其他参数依然按照程序默认的参数，通过以准确率作为评判标准，最终求出的结点划分最小样本数（min_samples_split）为 41，树的最大深度（max_depth）为 17，此时训练数据对应的准确率为 98.53%，模型的准确率有所提升，说明这两个参数的调优对模型准确率的提高有促进作用。

(3)叶子结点最小样本数（min_samples_leaf）的设置，这个参数和内部结点再划分所需要的最小样本数的作用类似，也是起到防止树过拟合，由于这个变量与内部结点再划分最小样本数有着较大的关系，因为二者有时候都会决定树是否要生成新的叶子，所以又把这两个变量进行了组合在一起进行参数的调优过程。在这步，树的棵数（n_estimators）依然设为 97，叶子结点最小样本数（min_samples_leaf）的寻找最优参数的区间设为[1, 10], 内部结点再划分所需要的最小样本数（min_samples_split）的寻找最优参数的区间设置为[41,45]，最高的准确率对应的参数值叶子结点最小样本数（min_samples_leaf）1，内部结点再划分所需要的最小样本数（min_samples_split）为 43，此时模型在训练样本上对应的准确率为 98.70%，模型在训练集上的预测准确率又有所提升。

（4）最大特征数（max_features）。训练单棵树时，每次从训练样本子集中随机选取多少个变量，变量选取的个数对每棵树的拟合效果有着一定的影响，变量选取个数太少，容易造成模型的欠拟合，单棵树利用数据的信息太少，而变量选取过多，树模型之间的差异也相对变小，导致树与树之间多样性变小，这样集成出来的效果也相对变小。关于其他需要设置的参数值，已经通过先前的步骤挑选出来，只需要保持先前的最优值不变即可，最大特征数的寻找最优值的区间设置为[8, 15]，通过比较模型的预测准确率，最终选取了最大特征数（max_features）为 14 的模型，此时模型对应准确率为 98.91%

综上所示，随机森林最终选定的最优参数分别为随机森林中树的个数（n_estimators）97，

内部结点再划分最小样本数（min_samples_split）43，最大深度（max_depth）14，叶子结点最小样本数（min_samples_leaf）1，最大特征数（max_features）14. 通过选取这些合适的参数，随机森林的效果也表现的较为优异，其中在训练样本上的准确率已经达到了 98.91%。

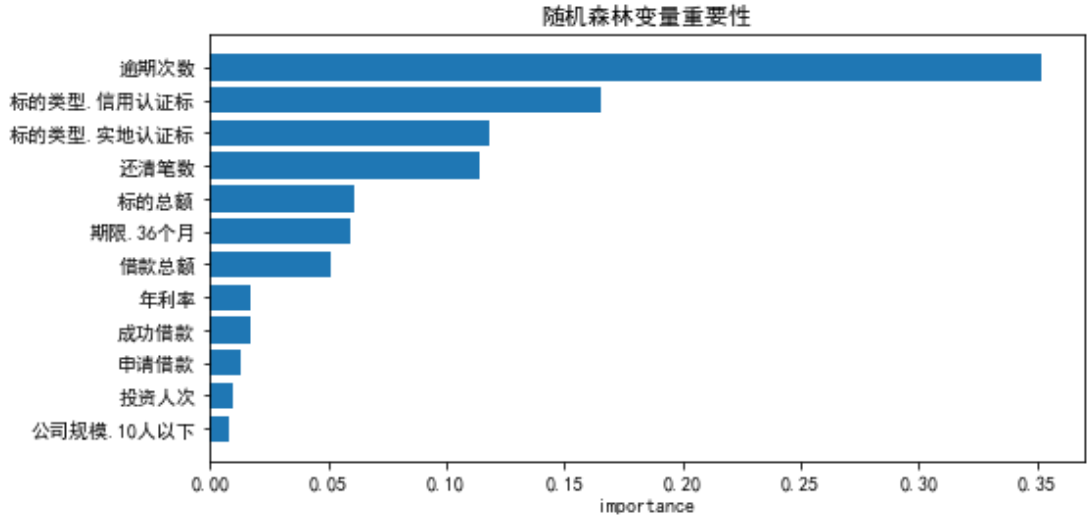

图 14 随机森林变量重要性

上图为随机森林模型输出的变量重要性示意图，从图中可以看出，随机森林模型中每个变量所起的作用都不太相同，有的变量重要性较大，有的变量重要性相对较小，其中逾期次数和还清笔数等变量在其中所起的作用比较重要，而成功借款次数和申请借款次数等变量在模型总所起的作用较小。

（三）XGBOOST 模型

1. XGBOOST 模型的训练及调参

虽然 xgboost 算法与随机森林都是基于树的集成，但是二者还是有着本质的不同，xgboost 算法在计算时每棵树是有相互联系的，后面的树基于前面树的结果进行生成，而随机森林每棵树是独立生成的，并且算法也希望每棵树要要保持相互之间的独立性。xgboost 的参数其中有一部分是和随机森林有相似的地方，另一部分则是独有的参数。

（1）xgboost 的迭代次数（n_estimators）：xgboost 模型需要建立多少个树函数，和随机森林建立树的个数的概念类似。首先对该参数进行调优，将其他待估计的参数首先设为默认值，让迭代次数从 50 到 100 逐渐变动，比较不同的迭代次数所对应的准确率值的大小，最终选择使得准确率最高的迭代次数，其中最优的模型的迭代次数（n_estimators）为 98，其对应的准确率为 98.10%

（2）最小叶子分数（min_child_weight）和树的深度（max_depth）：叶子样本分数和的最小值，如果小于该值，则不应该存在该节点，和决策树的叶子结点最小样本数类似，不过与决策树叶子结点最小样本数不同的是，该参数还考虑了每个叶子结点的对应的分数值，每个叶子内的样本数乘以对应的分数即为叶子样本分数和。树的深度表示新增的树在建立时最大的层数是多少，树太深容易造成过拟合。这两个参数都对树的结构和防止树过拟合有关，其中迭代次数（n_estimators）设置为 98，树的最大深度的可选择的区间为[4, 6]，最小叶

子分数的可选择区间设置为[1, 3]，其他参数设置为默认值，通过网格搜索组合这两个参数，其中准确率最高的模型的对应的参数分别为树的最大深度（max_depth）5，最小叶子分数（min_child_weight）为 1，此时模型的准确率的对应数值为 98.60%

（3）Gamma 参数：新增的树在确定其结构时，确定其是否增加结点的参数，如果新增加结点，能够使目标损失函数的减小值大于 Gamma，则增加结点。该参数越大，则算法越保守。Gamma 参数设置的寻优集合为{0,0.1,0.2,0.3,0.4}，最终通过比较模型准确率，得到参数的最优取值为 0，其对应的准确率为 98.66%

（4）Subsample 和 Colsample_bytree：Subsample 确定新增的树随机采样比例，和随机森林中选取的训练样本子集类似。Colsample_bytree 确定新增的树选取的变量个数，和随机森林每棵树选取变量类似。其中 Subsample 的寻找最优参数的区间设置为[0.60, 0.75]，Colsample_bytree 的寻找最优参数的区间设置为[0.75, 0.90]，通过比较不同参数组合对应的准确率，最终选定模型的参数为 Colsample_bytree0.85，Subsample0.65.

（5）Reg_alpha 和 reg_lambda：Reg_alpha 是目标损失函数 L1 正则化项的参数，用于控制树的结构。reg_lambda 是目标损失函数 L2 正则化项的参数，用于控制每个叶子的分数，防止过拟合。其中 Reg_alpha 的寻找最优参数的集合设置为{0,0.01,0.02,0.03}，reg_lambda 的寻找最优参数的集合设置为{0.3,0.5,0.7}，通过比较模型的准确率，选出最优参数 reg_lambda 为 0.5，Reg_alpha 为 0.此时模型对应的准确率为 99.10%

（6）由于前面已经更改了许多参数的取值，考虑到模型的参数之间会有一定的影响，对迭代次数（n_estimators）重新校准，校准区间为[80, 95]，依然根据准确率选择参数，选取的最优参数值为 86.

（7）学习率（learning_rate）：在新生成的树对应的函数并不直接和已求出的函数相加，需要乘以一个学习率，防止数据过拟合。保持其他参数不变，学习率的变动区间设置为{0.01,0.05,0.07,0.1,0.2}，比较不同参数对应的准确率，选择最优的学习率为 0.2，此时模型对应的准确率为 99.12%

经过一系列复杂的参数值的调整，最终确定了以下合适的参数值，它们分别为，树的个数(n_estimators)为 86,树的最大深度(max_depth)为 7,最小叶子分数(min_child_weight)为 2,随机选取样本数（subsample）为 0.65,随机选取变量数（colsample_bytre）为 0.85，控制节点生成的参数（gamma）为 0.2，两个正则化项参数（reg_alpha，reg_lambda）分别为 0 和 0.5，其中在这些参数控制下，xgboost 模型的表现非常优秀，在训练集上的准确率达到了 99.12%，可见 xgboost 模型的威力的确非同凡响，名不虚传。

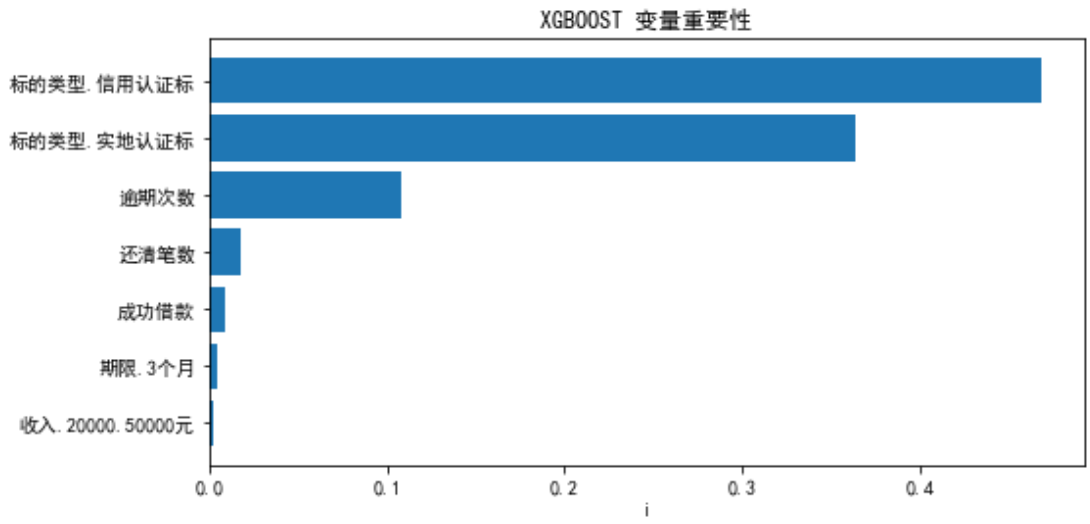

图 15 xgboost 变量重要性

上图为 xgboost 模型输出的变量示意图，可以看出每个变量对于 xgboost 模型的影响差别有明显不同，同时可以看出 xgboost 模型对少数变量的依赖性十分强，比如逾期次数和标的类型等变量，从输出结果中可以看出，这个变量的重要性远远超过了其他变量，以致于像学历、车产等变量的变量重要性几乎为 0。

（四）Stcking 集成模型

1. Stacking 算法简介

Stacking 算法的思想：

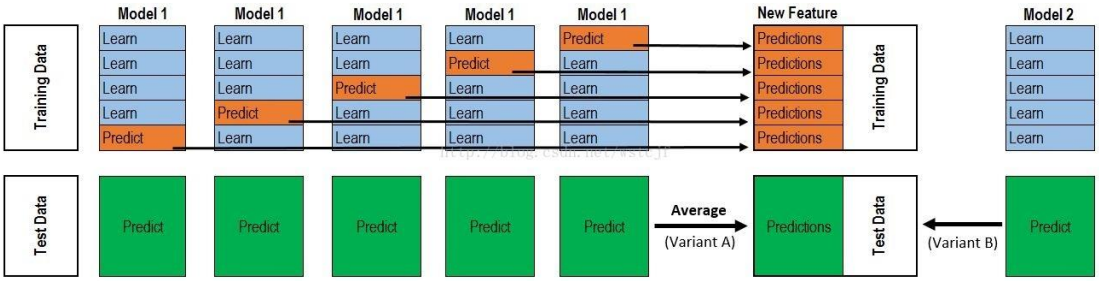

图 16 stacking 算法

图 14 为 stacking 算法的经典描述，本文将以该图为例介绍 stacking 算法。

第一层首先使用单个学习器对数据进行预测，第二层使用学习器根据前面得出的的判断概率和样本真值作为数据进行预测。

第一层：假如 Training Data 与 Test Data 数据分别为 1 万条和 3 千条。用 Model1 对 Training Data 进行 5 折交叉验证，将 Training Data 分为 5 部分，每部分有 2000 个数据，先利用前面 1,2,3,4 部分的 8000 个数据对 model1 进行训练，记得到的模型为

model11，然后利用 1,2,4,5 部分的 8000 个数据得到 model12，以此类推，可以得到 model13，model14，model15。接着利用 model11 对第五部分数据进行预测，得到第五部分的 2000 个预测值，利用 model12 对第四部分数据进行预测，得到第四部分的 2000 个预测值，以此类推，可以得到整个 Training Data 的预测值，记为 A1。同样可以利用 model11 至 model15，对初始测试集进行预测，并将预测值取平均，即可当得到 Test Data 的一组预测值，记为 B1。同理可利用 model2 产生 A2，B2，利用 model3 产生 A3，B3。将 A 类列合并，产生新的训练集，将 B 类列合并，产生新的测试集。

第二层：使用新的 Training Data 训练模型，使用新的 Test Data 验证模型。

算法的具体实现方式如下：

输入：数据集 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$

初级学习算法 $\phi_1,\phi_2,\cdots,\phi_T$

次级学习算法 ϕ

输出：集成算法 $H(x)$

流程：

1. for $t=1,2,\cdots,T$
   $h_t=\phi_t(D)$
   end for
2. 设$D'=\emptyset$
3. for $i=1,2,\cdots,m$
   for $t=1,2,\cdots,T$
   $z_{it}=h_t(x_i)$
   end for
   $D'=D'\cup D'((z_{i1},z_{i2},\cdots,z_{iT}),y_i)$
   end for
4. $h'=\phi(D')$
5. $H(x)=h'(h_1(x),h_2(x),\cdots,h_T(x))$

2. Stacking 集成模型的训练及调参

Stacking 方法通过通过引入带有l_2正则化项的 logitstic 模型把前面训练出来的含有最优参数的 KNN 算法模型、随机森林、xgboost 模型组合在一起，把 logistic 模型正则化项的系数的寻找最优系数的区间设为[1, 15]，步长为 1，通过多次迭代，以信用风险违约预测准确率作为选择参数的标准，确定最优的参数为 12，此时训练集上准确率为 99.466%

五、模型性能评估

（一）混淆矩阵

在测试集上检验如下指标：

表 4 不同模型在测试集上的表现

	准确率	查准率	查全率	F1
KNN 算法	91.403%	86.922%	89.686%	88.282%
随机森林	95.503%	99.419%	95.715%	97.532%
Xgboost	98.668%	92.860%	98.057%	95.279%
Stacking 集成	99.014%	99.812%	95.862%	97.797%

上表展示了本文所建立的四种模型的评价指标，1) KNN 算法的预测能力相对较差，因为其准确率为 91.403%，查准率为 86.922%，查全率为 89.686%，F1 值为 88.282%，四个指标都比其他模型低，尤其是查准率和 F1 值差距更明显。其他三个模型的表现都相对较好，每个指标的值都达到了 90%以上，说明这三个模型对于网贷平台信用风险的预测的精度和稳定性都很好。2) 随机森林和 xgboost 两个模型相比，都没有绝对的优势，xgboost 的准确率和查全率高于随机森林，而随机森林的查准率和 F1 高于 xgboost，二者各有所长。3) 综合比较四个模型，stacking 集成模型在各个方面都表现的比较优秀，对网贷平台风险的预测以及模型稳定性都是最好的。

（二）ROC 曲线

1. ROC 曲线与 AUC 值

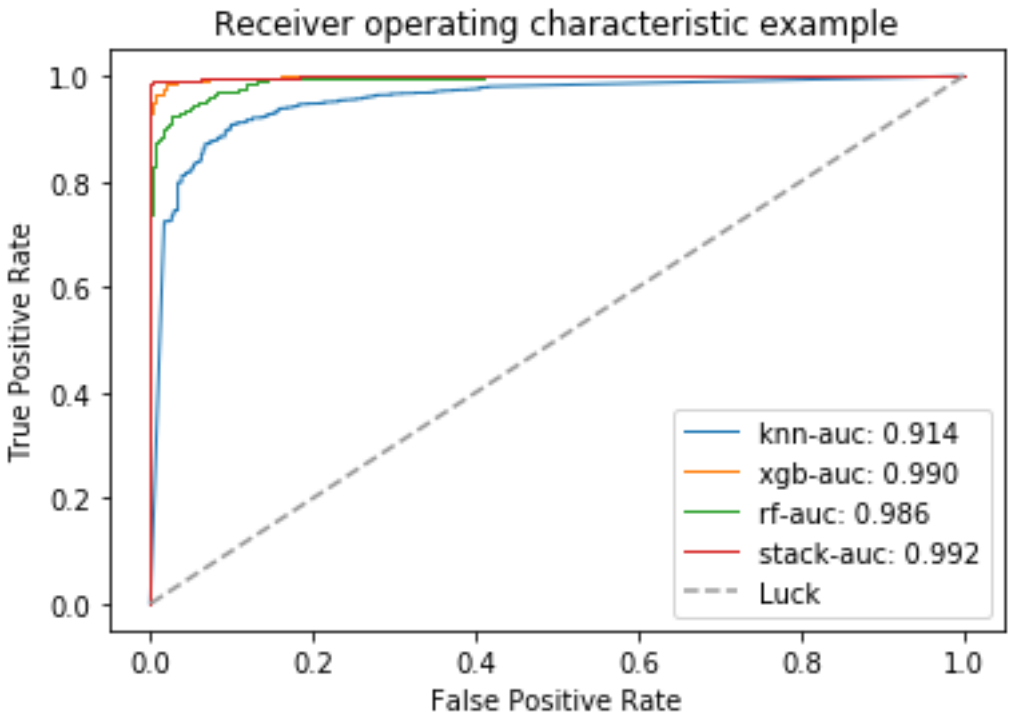

图 17 各模型的 ROC 曲线图

上图绘制出了四种模型的 ROC 曲线和 auc 值（1）从 ROC 曲线可以看出，KNN 模型的被其他三个模型完全包围，其在网贷平台风险预测方面性能相对较差，而 stacking 集成模型

比 xgboost 模型略好，将其他三个模型完全包围，说明 stacking 模型在网贷平台风险预测方面性能更胜一筹。（2）从 auc 的取值来看，KNN 算法的 auc 值为 0.914，在所有模型中表现最差，stacking 集成模型与 xgboost 模型的 auc 值同为 0.990，二者在网贷平台风险预测方面性能差不多。

从 ROC 曲线形状和 auc 值两方面来考虑，stacking 集成模型在网贷平台风险预测方面性能最好，xgboost，随机森林和 KNN 算法依次次之。

2. Lift 曲线

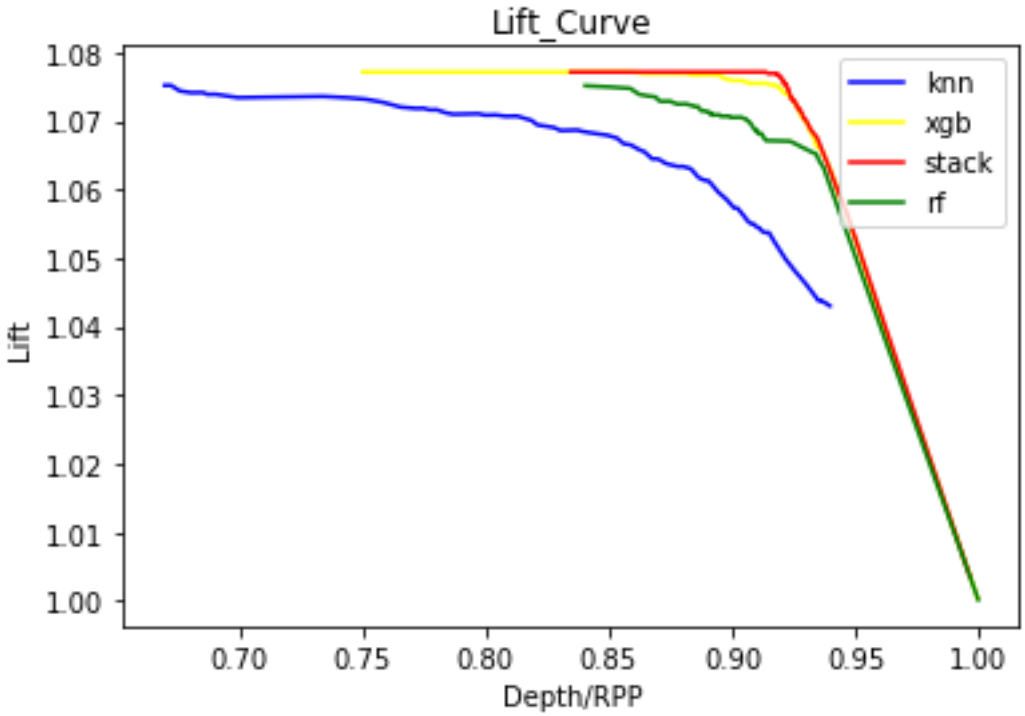

图 18 不同模型的 Lift 曲线

在模型对借款人信用风险的预测能力提升方面，KNN 依旧是表现的最差的模型，其对预测能力的提升明显比其他三个模型都低，随机森林稍好一些，但是其曲线仍在 xgboost 和 stacking 集成模型的下方，而 stacking 集成模型对应的 lift 曲线和 xgboost 的 lift 曲线交错在一起，二者的模型提升能力差不多。

从一些简单的评估指标、ROC 曲线、auc 值和 lift 图等方面的分析，可以发现模型对于网贷信用风险预测的综合性能方面，KNN 算法的表现最不好，而随机森林比 KNN 算法略好一些，xgboost 模型和 stacking 集成模型的在一些方面比较接近，但是 stacking 算法相对来说有一些优势，表现得最好。

结论

目前 P2P 网贷平台仍然存在者许多问题，尤其在 2018 年，P2P 平台的暴雷现象尤其这严重，这给众多的投资者带来了不小的损失。其中信用风险较高便是 P2P 网贷平台问题频出的重要原因之一，然而目前行业内并没有形成统一的信用风险衡量标准，国家也对于

P2P 网贷平台的正常健康发展高度重视，如何能够有效地减少信用风险的发生是 P2P 网贷平台面临的一大难题。

本文利用 stacking 方法融合了 KNN 算法、随机森林、xgboost 和 logistic 模型结合探索性分析结果，对人人贷平台的借款人的违约情况进行了预测。通过本文的分析过程可以发现以下结论：

1. 地域差别明显

从借款人的违约率的地理分布来看，西部以及西北部的省份的违约率稍高于东部和中部省份，同时东部成交量大大高于西部。这说明，东部沿海城市等发达城市信用经济发展较为活跃，适合 P2P 平台这类互联网金融部门拓展业务，开阔市场。

2. 高回报伴随高风险

利率较高、还款周期较短的标的的借款人违约率大大高于利润平缓、还款周期较长的标的的借款人，这充分表明了看似回报值较高的资产配置往往蕴藏着巨大风险，平台应该合理设置较为稳定的平台利率与适当长度还款周期，这有利于降低违约风险。对于借款标的利率设置偏高且又周期较短的用户重点监控，防止其发生违约行为。

3. 历史信用信息和是否认证对违约预测结果影响较大

从模型输出变量重要性来看，历史信用信息和是否认证对借款人违约情况有重大影响，逾期次数越多的借款人更可能违约，实地认证和信用认证之后的借款信用风险更小。平台应该更加重视信用历史数据的收集，同时积极配合线下认证，降低标的的违约风险。

4.本文依据人人贷平台的数据建立了预测借款人是否违约的模型，模型的精度和泛化能力都表现十分良好，对于借款人的信用风险评估可以起到一定帮助作用。

尽管本文对于网贷平台的信用风险的研究做了大量的工作，但是问题是无穷的的，人的能力是有限的。小组认为本文还存在以下局限和不足：1.本文建立的模型是基于现有的网贷数据所建立，一旦有新的网贷数据添加进来，无法进行动态调整，只能重新建立相关的信用风险预测模型。2.本文模型的预测结果的输出是一个 0-1 变量，只能预测借款人是否违约，而无法对借款人进行信用打分和评估借款人的信用额度等。3.鉴于时间和能力的因素，本文涉及的模型多为一些经典模型，并且没有尝试利用其它模型进行对比分析，无法证明本文的模型对网贷平台信用风险的预测一定是最优的。由此可见，本文仅做了网贷平台信用风险一小方面的研究，还有更广阔的天地值得探索。

参考文献

[1]迟国泰,潘明道,齐菲.一个基于小样本的银行信用风险评级模型的设计及应用[J].数量经济技术经济研究,2014,31(06):102-116.

[2]丁岚,骆品亮.基于 Stacking 集成策略的 P2P 网贷违约风险预警研究[J].投资研究,2017,36(04):41-54.

[3]古定威,丁岚,骆品亮.P2P 网贷平台信用风险控制的演化博弈分析[J].研究与发展管理,2018,30(03):12-21.

[4]黄大明.P2P 借贷平台风险与风险控制浅析[J].山西农经,2018(20):84-85.

[5]盛杰,刘岳,尹成语.基于多特征和 Stacking 算法的 Android 恶意软件检测方法[J].计算机系统应用,2018,27(02):197-201.

[6]王芳. 多特征融合的博客文章排序和分类算法研究[D].兰州理工大学,2012.

[7]王会娟,廖理.中国 P2P 网络借贷平台信用认证机制研究——来自“人人贷”的经验证据[J].中国工业经济,2014(04):136-147.

[8]向晖,杨胜刚. 消费者信用评估的 SVM-LDA 组合模型[J]. 消费经济, 2010,26(01):54-56.

[9]叶湘榕.P2P 借贷的模式风险与监管研究[J].金融监管研究,2014(03):71-82.

[10]张义娜. 中国 P2P 借贷模式的问题及对策研究[D].广西大学,2017.

[11]中国人民银行征信中心与金融研究所联合课题组. 互联网信贷、信用风险管理与征信[J]. 金融研究, 2014(10):133-147.

[12]周晓彧. 中小企业融资信用风险评级体系研究[D].天津工业大学,2016.

[13]中国 P2P 借贷服务行业白皮书[M]. 中国经济出版社 ，第一财经新金融研究中心，2013．

[14]Altman E I, Haldeman R G, Narayanan P, Zeta analysis: a new model to identify bankruptcy risk of corporations[J], Journal of Banking and Finance, 1977,1(2):29-54.

[15]Baesens B, T Van Gestel, S Viaene, M Stepanova, J Suykens, J Vanthienen, Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring[J], Journal of the Operational Research Society, 2003,54:627-635

[16]Bandyopadhyay A, Predicting probability of default of Indian corporate bonds: logistic and Z-score model approaches[J], The Journal of Risk Finance, 2006,7(3):255-272.

[17]Batista G, Prati R C, Monard W C, A study of the behavior of several methods for balancing machine learning training data[J], ACM Sigkdd Explorations Newsletters, 2004,6(1):20-29.

[18]Batuwita R, Palade V, FSVM-CIL: Fuzzy Support Vector Machines for Class Imbalance Learning[J], IEEE Transactions on Fuzzy Systems, 2010,18:558-571.

[19]Bauer, Eric, Kohavi, et al. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants[J]. Machine Learning, 1999, 36(1-2):105-139.

[20]Bhattacharyya S, Jha S, Tharakunnel K, Westland J.C.,Data Mining for Credit Card Fraud A Comparative Study[J], Decision Support Systems, 2011,50:602-613 .

[21]Blagus R, Lusa L, Class prediction for high-dimensional class-imbalanced data[J], BMC

Bioinformaatics, 2010,11:523.

[22]Boritz J E, Kennedy D B, Effectiveness of neural network types for prediction of business failure[J], Expert System with Applications, 1995,9(4):503-512.

[23]Craig W R, A factor analytic approach to bank condition[J], Journal of Banking and Finance, 1985,9:253-266.

[24]ELKAN C. The foundations of cost-sensitive learning[C]. Proceedings of the 17th International Joint Conference of Artificial Intelligence, Seattle, Washington, USA,2001:973-978

[25]Emekter Ｒ，Tu Y，Jirasakuldech B，et al．2014，Evaluating Credit Ｒisk and Loan Performance in Online Peer-to-Peer (P2P) Lending，Applied Economics，47(1) :54－70．

[26]Fan W, Stolfo S J, Zhang J, et al. AdaCost: Misclassification Cost-Sensitive Boosting[C], Proceedings of the 16th International Conference of Machine Learning, Bled, Slovenia, 1999:97-105.

[27]Fawcett T. An introduction to RPC analysis[J], Pattern Recognition Letters, 2006,27(8):861-874.

[28]Frohlich H, Chapelle O, Scholkopf B, Feature Selection for Support Vector Machines by Means of Genetic Algorithm[J], Proceedings of the 25th International Conferernce on Tools with Aritificial Intelligence, 2003:142-148.

[29]Galak J，Small D，Stephen A T，2011，Microfinance Decision Making: A Field Study of Prosocial Lending，Journal of Marketing Ｒesearch，48(SPL): 130－137

[30]García V, Marqués A I, Sánchez J S. Non-parametric Statistical Analysis of Machine Learning Methods for Credit Scoring[M], Management Intelligent Systems. 2012:263-272.

[31]Hand D J, Henley W E. Statistical Classification Methods in Consumer Credit Scoring: A Review[J]. Journal of the Royal Statistical Society, 1997, 160(3):523-541.

[32]HAN H, WANG W Y, MAO B H. Borderline-SOMTE: A New Over-Sampling Method in Imbalanced Data Sets Learning[C] Proceedings of the 2005 International Conference of Intelligent Computing, Hefei, China, 2005:878-887

[33]Harris T, Credit scoring using the clustered support vector machine[J], Expert Systems with Applications, 2015,42(2):741-750.

[34]Herzenstein M，Dholakia U M，Andrews Ｒ L， Strategic Herding Behavior in Peer-to-Peer Loan Auctions，Journal of Interactive Marketing. 2011,25:27－36．

[35]Huang G, Huang G B, Song S, et al. Trends in Extreme Learning Machine: A Review[J], Neural Networks, 2015.61:32-48.

[36]Jackson P, Perraudin W, Regulatory implications of credit risk modelling[J], Journal of Banking and Finance, 2000,24:1-14.

[37]JAPKOWICZ N, STEPHEN S. The class imbalance problem: A systematic study [J]. Intelligent Data Analysis, 2002,6:429-450.

[38]Li K, Kong X, Lu Z, et al. Boosting weighted ELM for imbalanced learning[J], Neurocomputing, 2014,128:15-21.

[39]Liu X Y, Wu J, Zhou Z H, Exploratory Undersampling for Class-Imbalanced Learning[J], IEEE Transactions on Systems, Man and Cybernetics, Part B,Cybernetics, 2009,39:539-550. [15] LOPEZ V, FERNANDEZ A, GARCIA S, et al. AN insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics [J]. Information Science, 2013,250:113-141.

[40]Lundy M, Cluster analysis in credit scoring, Credit scoring and credit control[M], New York: Oxford University Press, 1993,25-36.

[41]Mild A, Waitz M, Wckl J, How low can you go—Overcoming the inability of lenders to set proper interest rates on unsecured peer-to-peer lending markets[J], Journal of Business Research, 2015,68(6):1291-1305.

[42]Platt J, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods[J], Advances in large margin classifiers, 1999,10(3):61-74.

[43]SEIFFERT C, KHOSHGOFTAAR T M, HULSE VANJ. Hybrid sampling for imbalanced data[J]. Integrated Computer-Aided Engineering,2009,16(3):193-210

[44]Sun Y, Kamel M S, Wong A K C, et al. Cost-Sensitive Boosting for Classification of Imbalanced Data[J], Pattern Recognition, 2007,40(12):3358-3378.

[45]Tsai C F, Chen M L, Credit rating by hybrid machine learning techniques[J], Applied soft computing, 2010,10(2):374-380.

[46]Twala B. Multiple classifier application to credit risk assessment[J]. Expert Systems with Applications, 2010, 37(4):3326-3336

[47]Vapnik V N, The nature of statistical learning theory[M], New York: Spring-Verlag, 1995.

[48]Veropoulos K, Campbell C, Creitianini N, Controlling the sensitivity of support vector machines[C], Proceedings of the International Joint Conference of Artificial Intelligence, 1999:55-60.

[49]Wang G, Hao J, Ma J,Jiang H, A Comparative Assessment of Ensemble Learning for Credit Behavior[J]Journal of Financial and Quantitative Analysis, 2011,15:757-770.

[50]Yao P. Credit Scoring Using Ensemble Machine Learning[C]// International Conference on Hybrid Intelligent Systems. IEEE, 2009:244-246.

[51]Yeh I C, Lien C H. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients[J]. Expert Systems with Applications, 2009, 36(2):2473-2480.

[52]YU H, SUN C, YANG X, et al. ODOC-ELM: Optimal decision outputs compensation-based extreme learning machine for classifying imbalanced data[J]. Knowledge-Based Systems, 2016,92:55-70

[53]Zenko B, Todorovski L, Dzeroski S. A Comparison of Stacking with Meta Decision Trees to Bagging, Boosting, and Stacking with other Methods[C], IEEE International Conference on Data Mining. IEEE Computer Society, 2001:669-670.

[54]ZHOU Z H, LIU X Y. On Mult-i class Cost Sensitive Learning[J]. Computational

Intelligence,2010,26(3):232-257

[55]Zong W, Huang G B, Chen Y, Weighted extreme learning machine for imbalabce learning[J], Neuro-computing, 2013,101:229-242.

附件

附件 A（Python 爬虫代码）

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException
from bs4 import BeautifulSoup
import requests
import pandas as pd
import numpy as np
import time
from numpy import nan as NA

def search():
    '''
    模拟登陆
    '''
    try:
        browser.get("https://www.renrendai.com/login")
        password_login = waits.until(EC.element_to_be_clickable((By.CSS_SELECTOR, "#form-login > div > div.forget-password > span")))
        browser.execute_script("$(arguments[0]).click()", password_login)

        # 等待用户名和密码框的出现
        user_name = waits.until(EC.presence_of_element_located((By.CSS_SELECTOR, "#login_username")))
        password = waits.until(EC.presence_of_element_located((By.CSS_SELECTOR, "#J_pass_input")))
        # 等待登录按钮可点击
        submit = waits.until(EC.element_to_be_clickable((By.CSS_SELECTOR, "#form-reg > div > div.button-block > button")))

        # 用户名和密码框的内容清除
        user_name.clear()
        password.clear()
        # 用户名和密码的输入
        user_name.send_keys("")
        password.send_keys("")
        # 点击登录按钮
        browser.execute_script("$(arguments[0]).click()", submit)
        # 确保登陆成功，以登录成功后页面内出现“充值”元素出现为准
        waits.until(EC.presence_of_element_located((By.CSS_SELECTOR, "#p2p-index > div > ul > li.p2p-btn.fn-clear > a.p2p-recharge")))
        # submit.click()
```

```
    # 若出现超时按钮则重新执行方法
    except TimeoutException:
        search()

def get_html(url):
    '''
    获取页面 html
    '''
    try:
        browser.get(url)
        # 等待页面某个元素出现，确保页面加载成功
        waits.until(EC.presence_of_element_located((By.CSS_SELECTOR, "#loan-transfer-detail > div > div.loan-content > div.loan-con-l > div.loan-l-number > p.w285")))

        # 获取 html
        html = browser.execute_script("return document.documentElement.outerHTML")

        # 若 html 为 none，则重新获取
        if html is None:
            waits.until(EC.presence_of_element_located((By.CSS_SELECTOR, "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(3) > li.w37.t-i-14 > span.other-color")))
            html = browser.page_source
        return(html)
    except TimeoutException:
        get_html(url)


def get_one_num(html, selector):
    '''
    从 html 中获取元素的值
    '''
    try:
        soup = BeautifulSoup(html, "lxml")
    except:
        soup = None
    try:
        # pd.Series 有利于后面合并为数据框
        var = pd.Series(soup.select(selector)[0].get_text().strip())
    except:
        var = NA
    return (var)
```

```
def get_one_page(html):
    '''
    获取每个样本的信息
    '''
    try:
        soup = BeautifulSoup(html, "lxml")
    except:
        soup = None

    try:
        # 获取每个样本对应的 state，还款中，还清，垫付
        state = pd.Series(soup.select("#loan-transfer-detail > div > div.loan-content > div.loan-content-right-desc > div.loan-box-top > em")[0].get_text().strip())
    except:
        state = "unknown"
    try:
        # 获取与上面变量相对应的变量
        advance_payment = pd.Series(soup.select(
                "#loan-transfer-detail > div > div.loan-content > div.loan-content-right-desc > div.loan-box-top > p > span")[0].get_text().strip())
    except:
        advance_payment = "unknown"


    string = ["垫付金额", "还清时间"]
    # 确保获取的数据均是还清或垫付类型
    if (state[0] in string) and (advance_payment[0] != "已流标"):

        title = get_one_num(html, "#loan-transfer-detail > div > div.loan-agreement > p")

        total_amount = get_one_num(html,
            "#loan-transfer-detail > div > div.loan-content > div.loan-con-l > div.loan-l-number > p.w285")

        year_rate = get_one_num(html,
            "#loan-transfer-detail > div > div.loan-content > div.loan-con-l > div.loan-l-number > p.w205")

        repayment_term = get_one_num(html,
            "#loan-transfer-detail > div > div.loan-content > div.loan-con-l > div.loan-l-number > p.w150")

        prepayment_rate = get_one_num(html,
            "#loan-transfer-detail > div > div.loan-content > div.loan-con-l > ul > li:nth-of-type(2) > span")

        repayment_way = get_one_num(html,
            "#loan-transfer-detail > div > div.loan-content > div.loan-con-l > ul > li:nth-of-type(3) > span")
```

```
        repayment_source = get_one_num(html,
            "#loan-transfer-detail > div > div.loan-content > div.loan-con-l > ul > li:nth-of-type(4) > span")

        credit_grade = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(1) > li:nth-of-type(2) > em > i")

        age = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(3) > li.w37.t-i-14 > span.other-color")

        education = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(3) > li:nth-of-type(2) > span.other-color")

        marriage = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(3) > li.w26 > span.other-color")

        application_loan = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(5) > li.w37.t-i-14.li-p-b > span.other-color > span:nth-of-type(1)")

        success_loan = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(6) > li.w37.t-i-14.li-p-t.li-p-b > span.other-color > span:nth-of-type(1)")

        repaid = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(7) > li.w37.t-i-14.li-p-t > span.other-color > span:nth-of-type(1)")

        credit_amount = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(5) > li:nth-of-type(2) > span.other-color > span:nth-of-type(1)")

        loan_amount = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(6) > li:nth-of-type(2) > span.other-color > span:nth-of-type(1)")

        not_return_amount = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(7) > li:nth-of-type(2) > span.other-color > span:nth-of-type(1)")
```

```
        overdue_money = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(5) > li.w26.li-p-b > span.other-color > span:nth-of-type(1)")

        overdue_num = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(6) > li.w26.li-p-t.li-p-b > span.other-color > span:nth-of-type(1)")

        serious_overdue = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(7) > li.w26.li-p-t > span.other-color > span:nth-of-type(1)")


        income = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(9) > li.w37.t-i-14.li-p-b > span.other-color")

        house = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(9) > li:nth-of-type(2) > span.other-color")

        house_loan = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(9) > li.w26.li-p-b > span.other-color")

        car = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(10) > li.w37.t-i-14.li-p-t > span.other-color")

        car_loan = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(10) > li:nth-of-type(2) > span.other-color")

        others = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(10) > li.w26.li-p-t-other > span.other-color.other-info")


        company = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(12) > li.w37.t-i-14.li-p-b > span.other-color")
        company_scale = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(12) > li:nth-of-type(2) > span.other-color")
```

```
        job = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(12) > li.w26.li-p-b > span.other-color")

        work_province = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(13) > li.w37.t-i-14.li-p-t > span.other-color > span:nth-of-type(1)")

        work_city = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(13) > li.w37.t-i-14.li-p-t > span.other-color > span:nth-of-type(3)")

        work_time = get_one_num(html,
            "#lend-loan > div > div > div.wdt-lend-info > div > div.borrower-info > ul:nth-of-type(13) > li:nth-of-type(2) > span.other-color")

        dataset = pd.DataFrame(
            {"title": title, "total_amount": total_amount, "year_rate": year_rate, "repayment_term": repayment_term,
             "prepayment_rate": prepayment_rate,
             "repayment_way": repayment_way, "repayment_source": repayment_source, "credit_grade": credit_grade, "age": age,
             "education": education,
             "marriage": marriage, "application_loan": application_loan, "success_loan": success_loan, "repaid": repaid,
             "credit_amount": credit_amount,
             "loan_amount": credit_amount, "not_return_amount": not_return_amount, "overdue_money": overdue_money,
             "overdue_num": overdue_num, "serious_overdue": serious_overdue,
             "income": income, "house": house, "house_loan": house_loan, "car": car, "car_loan": car_loan,
             "others": others, "company": company, "company_scale": company_scale, "job": job,
             "work_province": work_province,
             "work_city": work_city, "work_time": work_time, "state": state, "advance_payment": advance_payment}, index=[0])
    else:
        dataset = "unmeaningful"
    return(dataset)

p2p = pd.DataFrame({"title":["a"], "total_amount":["a"], "year_rate":["a"], "repayment_term":["a"], "prepayment_rate":["a"],
                    "repayment_way":["a"], "repayment_source":["a"], "credit_grade":["a"], "age":["a"], "education":["a"],
                    "marriage":["a"], "application_loan":["a"], "success_loan":["a"], "repaid":["a"],
```

```
"credit_amount":["a"],
                        "loan_amount":["a"],        "not_return_amount":["a"],        "overdue_money":["a"],
"overdue_num":["a"], "serious_overdue":["a"],
                        "income":["a"], "house":["a"], "house_loan":["a"], "car":["a"], "car_loan":["a"],
                        "others":["a"],        "company":["a"],        "company_scale":["a"],        "job":["a"],
"work_province":["a"],
                        "work_city":["a"], "work_time":["a"], "state":["a"], "advance_payment":["a"]},index = [0])

url0 = "https://www.renrendai.com/loan-"
# 设置代理
chromeOptions = webdriver.ChromeOptions()
optionUrl = "--proxy-server=http://36.99.17.52:80"
chromeOptions.add_argument(optionUrl)
browser = webdriver.Chrome(chrome_options = chromeOptions)
# 检查代理是否成功
browser.get("http://httpbin.org/ip")
# browser.get("https://www.baidu.com")
print(browser.page_source)
waits = WebDriverWait(browser, 10)
def main(start, stop, dataset = p2p):
    search()
    # 翻页操作
    for i in range(start,stop):
        url = url0 + str(2000000 + i) + ".html"
        html = get_html(url)
        one_page = get_one_page(html)
        if one_page is None:
            print("jump")
        else:
            if len(one_page)==1:
                frames = [dataset,one_page]
                dataset = pd.concat(frames)
        if(i % 200 == 0):
            time.sleep(150)

    return(dataset)

p2p_data = main(7000, 7500)


browser.quit()


p2p_data = p2p_data[["title", "total_amount", "year_rate", "repayment_term", "prepayment_rate",
```

```
                    "repayment_way", "repayment_source", "credit_grade", "age", "education",
                    "marriage", "application_loan", "success_loan", "repaid", "credit_amount",
                    "loan_amount", "not_return_amount", "overdue_money", "overdue_num",
"serious_overdue",
                    "income", "house", "house_loan", "car", "car_loan",
                    "others", "company", "company_scale", "job", "work_province",
                    "work_city", "work_time", "state", "advance_payment"]]
p2p_data
p2p_data.to_csv("D:\Desktop\p2p.csv", encoding = "gbk")
```

附件 B（模型构建代码）

```
# KNN
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

train = pd.read_csv('train.csv', encoding='gbk')
test = pd.read_csv('test.csv', encoding='gbk')

X_train, y_train = train.drop(['标的状态.还清'], axis=1), train['标的状态.还清']
X_test, y_test = test.drop(['标的状态.还清'], axis=1), test['标的状态.还清']

params_grid = [
    {
        'weights':['uniform'],
        'n_neighbors':[i for i in range(5, 15)]
    },
    {
        'weights':['distance'],
        'n_neighbors':[i for i in range(5, 15)],
        'p':[i for i in range(1, 6)]
    }
]

knn = KNeighborsClassifier()
grid_search = GridSearchCV(knn, params_grid,
                           scoring='accuracy',
```

```
                                    cv=10,
                                    n_jobs=-1)
optimized_knn = grid_search.fit(X_train, y_train)
print('Train accuracy: %.5f' % optimized_knn.best_score_)    # validation accuracy best
print(optimized_knn.best_params_)

#  看在测试集上的效果
best_knn = optimized_knn.best_estimator_
best_knn.fit(X_train, y_train)
print('Test accuracy: %.5f' % best_knn.score(X_test, y_test))
best_knn.predict(X_test)

from sklearn import    metrics
metrics.precision_score(y_test,best_knn.predict(X_test))
metrics.precision_score(y_train,best_knn.predict(X_train))
metrics.recall_score(y_test, best_knn.predict(X_test))
# metrics.recall_score(y_train,best_knn.predict(X_train))

metrics.f1_score(y_test, best_knn.predict(X_test))
metrics.f1_score(y_train,best_knn.predict(X_train))

##  结果可视化



import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

#确认训练集的边界
x_min, x_max = X_test['年龄'].min(), X_test['年龄'].max()
y_min, y_max = X_test['年利率'].min(), X_test['年利率'].max()

# Create color maps
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA']) #给不同区域赋以颜色

Z = best_knn.predict(X_test)
Z = Z.reshape(X_test['年龄'].shape)
```

```
plt.scatter(X_test['年龄'], X_test['年利率'], c=Z, cmap=cmap_light)
plt.xlim(x_min-1, x_max+1)
plt.ylim(y_min-1, y_max+1)
plt.xlabel(u'年龄')
plt.ylabel(u'年利率')      # 可以使用中文，但需要导入一些库即字体
plt.title(u'KNN 模型预测结果一览')
plt.show()

# RandomForest
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import    metrics # cross_validation,

import matplotlib.pylab as plt

get_ipython().magic('matplotlib inline')
train = pd.read_csv('train.csv', encoding='gbk')
test = pd.read_csv('test.csv', encoding='gbk')
X_train, y_train = train.drop(['标的状态.还清'], axis=1), train['标的状态.还清']
X_test, y_test = test.drop(['标的状态.还清'], axis=1), test['标的状态.还清']

## n_estimators
cv_params = {'n_estimators': range(95,100,1)}
other_params = {'min_samples_split': 100, 'min_samples_leaf': 20, 'max_depth': 8,
                    'max_features': 'sqrt' , 'random_state': 10}
model = RandomForestClassifier(**other_params)
optimized_rf = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=4)
optimized_rf.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_rf.param_grid))
print('参数的最佳取值：{0}'.format(optimized_rf.best_params_))
print('最佳模型得分:{0}'.format(optimized_rf.best_score_))

## min_samples_split 及 max_depth
cv_params = {'max_depth':range(10,20,1), 'min_samples_split':range(40,45,1)}
other_params = {'n_estimators': 96, 'min_samples_split': 100, 'min_samples_leaf': 20,
```

```
                    'max_depth': 8, 'max_features': 'sqrt' , 'random_state': 10}

model = RandomForestClassifier(**other_params)
optimized_rf = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=4)
optimized_rf.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_rf.param_grid))
print('参数的最佳取值：{0}'.format(optimized_rf.best_params_))
print('最佳模型得分:{0}'.format(optimized_rf.best_score_))
## min_samples_split 和 min_samples_leaf
cv_params = {'min_samples_leaf': range(1,10,1), 'min_samples_split':range(40,45,1)}
other_params = {'n_estimators': 96, 'min_samples_split': 41, 'min_samples_leaf': 20,
                    'max_depth': 17, 'max_features': 'sqrt' , 'random_state': 10}
model = RandomForestClassifier(**other_params)
optimized_rf = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=4)
optimized_rf.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_rf.param_grid))
print('参数的最佳取值：{0}'.format(optimized_rf.best_params_))
print('最佳模型得分:{0}'.format(optimized_rf.best_score_))

## max_features
cv_params = {'max_features': range(8,15,1)}
other_params = {'n_estimators': 97, 'min_samples_split': 43, 'min_samples_leaf': 1,
                    'max_depth': 14, 'max_features': 'sqrt' , 'random_state': 10}
model = RandomForestClassifier(**other_params)
optimized_rf = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=-1)
optimized_rf.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_rf.param_grid))
print('参数的最佳取值：{0}'.format(optimized_rf.best_params_))
print('最佳模型得分:{0}'.format(optimized_rf.best_score_))

# 看在测试集上的效果
best_rf = optimized_rf.best_estimator_
best_rf.fit(X_train, y_train)
print('Test accuracy: %.5f' % best_rf.score(X_test, y_test))
metrics.accuracy_score(y_train,best_rf.predict(X_train))
```

```
metrics.precision_score(y_test,best_rf.predict(X_test))
metrics.precision_score(y_train,best_rf.predict(X_train))
metrics.recall_score(y_test,best_rf.predict(X_test))
metrics.recall_score(y_train,best_rf.predict(X_train))
metrics.f1_score(y_test,best_rf.predict(X_test))
metrics.f1_score(y_train,best_rf.predict(X_train))

## 每颗决策树展示
Estimators = best_rf.estimators_

#from IPython.display import Image
#from sklearn import tree
#import pydotplus
#import os
#
#Estimators = best_rf.estimators_
#for index, model in enumerate(Estimators):
#       filename = 'result_' + str(index) + '.pdf'
#       dot_data = tree.export_graphviz(model , out_file=None,
#                                                       feature_names=X_test.columns,
#                                                       class_names=y_test.name,
#                                                       filled=True, rounded=True,
#                                                       special_characters=True)
#       graph = pydotplus.graph_from_dot_data(dot_data)
#       # 使用 ipython 的终端 jupyter notebook 显示。
#       Image(graph.create_png())
#       graph.write_pdf(filename)

## 变量重要性
from itertools import product
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

importances = best_rf.feature_importances_
```

```
features = X_test.columns[importances>0.005]
importances = importances[importances>0.005]

indices = np.argsort(importances)
y_pos = np.arange(len(indices))

#  横向柱状图
plt.figure(figsize=(8,4))
plt.barh(y_pos, importances[indices], align='center')
plt.yticks(y_pos, [features[i] for i in indices])
plt.xlabel('importance')
plt.title('随机森林变量重要性')
plt.show()

# XGBOOST
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_breast_cancer
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pandas as pd

train = pd.read_csv('train.csv', encoding='gbk')
test = pd.read_csv('test.csv', encoding='gbk')
X_train, y_train = train.drop(['标的状态.还清'], axis=1), train['标的状态.还清']
X_test, y_test = test.drop(['标的状态.还清'], axis=1), test['标的状态.还清']

## n_estimators
cv_params = {'n_estimators': range(95,100,1)}
other_params = {'learning_rate': 0.1, 'n_estimators': 500, 'max_depth': 5, 'min_child_weight': 1, 'seed': 0,
                'subsample': 0.8, 'colsample_bytree': 0.8, 'gamma': 0, 'reg_alpha': 0, 'reg_lambda': 1}

model = XGBClassifier(**other_params)
optimized_GBM = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=4)
optimized_GBM.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_GBM.param_grid))
print('参数的最佳取值：{0}'.format(optimized_GBM.best_params_))
```

```
print('最佳模型得分:{0}'.format(optimized_GBM.best_score_))

## min_child_weight 及 max_depth
cv_params = {'max_depth': range(4,6), 'min_child_weight': range(1,3)}
other_params = {'learning_rate': 0.1, 'n_estimators': 98, 'max_depth': 5, 'min_child_weight': 1, 'seed': 0,
                    'subsample': 0.8, 'colsample_bytree': 0.8, 'gamma': 0, 'reg_alpha': 0, 'reg_lambda': 1}
model = XGBClassifier(**other_params)
optimized_GBM = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=4)
optimized_GBM.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_GBM.param_grid))
print('参数的最佳取值：{0}'.format(optimized_GBM.best_params_))
print('最佳模型得分:{0}'.format(optimized_GBM.best_score_))

## gamma

cv_params = {'gamma': [i / 10.0 for i in range(0,5)] }
other_params = {'learning_rate': 0.1, 'n_estimators': 98, 'max_depth': 5, 'min_child_weight': 1, 'seed': 0,
                    'subsample': 0.8, 'colsample_bytree': 0.8, 'gamma': 0, 'reg_alpha': 0, 'reg_lambda': 1}
model = XGBClassifier(**other_params)
optimized_GBM = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=-1)
optimized_GBM.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_GBM.param_grid))
print('参数的最佳取值：{0}'.format(optimized_GBM.best_params_))
print('最佳模型得分:{0}'.format(optimized_GBM.best_score_))

## subsample 和 colsample_bytree
cv_params = {'subsample': [i / 100.0 for i in range(60,75,5)],
                'colsample_bytree': [i / 100.0 for i in range(75,90,5)] }
other_params = {'learning_rate': 0.1, 'n_estimators': 98, 'max_depth': 5, 'min_child_weight': 1, 'seed': 0,
                    'subsample': 0.8, 'colsample_bytree': 0.8, 'gamma': 0, 'reg_alpha': 0, 'reg_lambda': 1}
model = XGBClassifier(**other_params)
optimized_GBM = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=-1)
optimized_GBM.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_GBM.param_grid))
print('参数的最佳取值：{0}'.format(optimized_GBM.best_params_))
```

```
print('最佳模型得分:{0}'.format(optimized_GBM.best_score_))

## reg_alpha 和 reg_lambda
cv_params = {'reg_alpha': [0, 0.01, 0.02, 0.03], 'reg_lambda': [0.3, 0.5, 0.7]}
other_params = {'learning_rate': 0.1, 'n_estimators': 98, 'max_depth': 5, 'min_child_weight': 1, 'seed': 0,
                'subsample': 0.65, 'colsample_bytree': 0.85, 'gamma': 0, 'reg_alpha': 0, 'reg_lambda': 1}
model = XGBClassifier(**other_params)
optimized_GBM = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=-1)
optimized_GBM.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_GBM.param_grid))
print('参数的最佳取值：{0}'.format(optimized_GBM.best_params_))
print('最佳模型得分:{0}'.format(optimized_GBM.best_score_))

### 重新校准 n_estimators
cv_params = {'n_estimators': range(80,95,1)}
other_params = {'learning_rate': 0.1, 'n_estimators': 98, 'max_depth': 7, 'min_child_weight': 2, 'seed': 0,
                'subsample': 0.65, 'colsample_bytree': 0.85, 'gamma': 0.2, 'reg_alpha': 0, 'reg_lambda': 0.5}
model = XGBClassifier(**other_params)
optimized_GBM = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=-1)
optimized_GBM.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_GBM.param_grid))
print('参数的最佳取值：{0}'.format(optimized_GBM.best_params_))
print('最佳模型得分:{0}'.format(optimized_GBM.best_score_))
## learning_rate
cv_params = {'learning_rate': [0.05, 0.1, 0.2]}
other_params = {'learning_rate': 0.1, 'n_estimators': 86, 'max_depth': 7, 'min_child_weight': 2, 'seed': 0,
                'subsample': 0.65, 'colsample_bytree': 0.85, 'gamma': 0.2, 'reg_alpha': 0, 'reg_lambda': 0.5}

model = XGBClassifier(**other_params)
optimized_GBM = GridSearchCV(estimator=model, param_grid=cv_params, scoring='accuracy', cv=10, verbose=1,
n_jobs=-1)
optimized_GBM.fit(X_train, y_train)
print('参数网格：{0}'.format(optimized_GBM.param_grid))
print('参数的最佳取值：{0}'.format(optimized_GBM.best_params_))
print('最佳模型得分:{0}'.format(optimized_GBM.best_score_))
```

```
# 看在测试集上的效果
best_GBM = optimized_GBM.best_estimator_
best_GBM.fit(X_train, y_train)
print('Test accuracy: %.5f' % best_GBM.score(X_test, y_test))


metrics.accuracy_score(y_train, best_GBM.predict(X_train))


metrics.precision_score(y_test, best_GBM.predict(X_test))


metrics.precision_score(y_train, best_GBM.predict(X_train))


metrics.recall_score(y_test, best_GBM.predict(X_test))

metrics.recall_score(y_train, best_GBM.predict(X_train))

metrics.f1_score(y_test, best_GBM.predict(X_test))

metrics.f1_score(y_train, best_GBM.predict(X_train))


#-*- coding: utf-8 -*-
from itertools import product
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

importances = best_GBM.feature_importances_
features = X_test.columns[importances>0.0015]
importances = importances[importances>0.0015]

indices = np.argsort(importances)
```

```
y_pos = np.arange(len(indices))

# 横向柱状图
plt.figure(figsize=(8,4))
plt.barh(y_pos, importances[indices], align='center')
plt.yticks(y_pos, [features[i] for i in indices])
plt.xlabel('i')
plt.title('XGBOOST 变量重要性')
plt.show()


# Stacking

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from mlxtend.classifier import StackingClassifier
from sklearn import model_selection
import numpy as np
import pandas as pd
import warnings; warnings.simplefilter('ignore')

train = pd.read_csv('train.csv', encoding='gbk')
test = pd.read_csv('test.csv', encoding='gbk')

X_train, y_train = train.drop(['标的状态.还清'], axis=1), train['标的状态.还清']
X_test, y_test = test.drop(['标的状态.还清'], axis=1), test['标的状态.还清']

clf1 = optimized_knn.best_estimator_
clf2 = optimized_rf.best_estimator_
clf3 = optimized_GBM.best_estimator_

lr = LogisticRegression(penalty='l1', C = 0.1)

sclf = StackingClassifier(classifiers=[clf1, clf2, clf3],
                                        use_probas=True,
                                        average_probas=False,
                                        meta_classifier=lr)
```

```
print('5-fold cross validation:\n')

for clf, label in zip([clf1, clf2, clf3, sclf],
                      ['KNeighborsClassifier',
                       'RandomForestClassifier',
                       'xgb.XGBClassifier',
                       'StackingClassifier',
                      ]):
    scores = model_selection.cross_val_score(clf, X_train, y_train,
                                             cv=10, scoring='accuracy')
    print("accuracy: %0.5f [%s]"
          % (scores.mean(), label))

params = { 'meta-logisticregression__C': range(10,15)}

grid = GridSearchCV(estimator=sclf,
                    param_grid=params,
                    scoring='accuracy', cv=10, verbose=1, n_jobs=-1)
grid.fit(X_train, y_train)

print('Best parameters: %s' % grid.best_params_)
print('accuracy: %.5f' % grid.best_score_)

# 看在测试集上的效果
best_stack = grid.best_estimator_
best_stack.fit(X_train, y_train)
print('Test accuracy: %.5f' % best_stack.score(X_test, y_test))
metrics.accuracy_score(y_train,best_stack.predict(X_train))
metrics.precision_score(y_test, best_stack.predict(X_test))
metrics.precision_score(y_train,best_stack.predict(X_train))
metrics.recall_score(y_test, best_stack.predict(X_test))
metrics.recall_score(y_train,best_stack.predict(X_train))
metrics.f1_score(y_test, best_stack.predict(X_test))
metrics.f1_score(y_train,best_stack.predict(X_train))

# 模型评价
import matplotlib.pyplot as plt
```

```
import numpy as np
from sklearn import metrics

class binary_evaluation:
    """二分类模型评估类
    输入:
        y_true:一维 np.array 对象，实际值
        y_pred:一维 np.array 对象，预测概率
        precision:bool，True 时 cutoff 阈值由预测概率逐一产生(速度慢，精度高)，False 时 cutoff 阈值分段产生(速度快，精度低)，默认 False
    方法：
        confusion_matrix(): 计算混淆矩阵的指标
        threhold(): 产生 cut-off 阈值
        index(): 计算不同 cut-off 阈值下的指标
        plot_roc(): 绘制 ROC 曲线
        plot_gains(): 绘制收益曲线，常用于营销类二分类模型
        plot_lift(): 绘制提升曲线，常用于营销类二分类模型
        plot_lorenz(): 绘制洛伦兹曲线，常用于信用风险建模
        plot_ks(): 绘制 ks 曲线，常用于信用风险建模
        plot_afdr_adr(): 绘制 afdr_adr 曲线，常用于欺诈侦测模型
        plot_precision_recall(): 绘制精准率-召回率曲线，功能与 ROC 曲线类似，但在样本不平衡时，ROC 曲线更佳稳健
    """

    def __init__(self,y_true,y_pred,precision=False):

        self.true = y_true
        self.pred = y_pred
        self.precision = precision


    def confusion_matrix(self,pred):

        # 产生混淆矩阵的四个指标
        tn, fp, fn, tp = metrics.confusion_matrix(self.true,pred).ravel()

        # 产生衍生指标
        fpr = fp/(fp + tn) #假真率／特异度
```

```
        tpr = tp/(tp + fn) #灵敏度 / 召回率
        depth = (tp + fp)/(tn+fp+fn+tp) #Rate of positive predictions.
        ppv = tp/(tp + fp) #精准率
        lift = ppv/((tp + fn)/(tn+fp+fn+tp)) #提升度
        afdr = fp/tp #(虚报 / 命中) / 好账户误判率
        return(fpr,tpr,depth,ppv,lift,afdr)


def threhold(self):

        min_score = min(self.pred) #预测概率最大值
        max_score = max(self.pred) #预测概率最小值

         # 精度要求下产生分阈值
        if self.precision is True:
            thr = self.pred
        elif self.precision is False:
            thr = np.linspace(min_score, max_score,100)
        else:
            raise ValueError('precision should be True or False!')

        return(thr)


def index(self):
        """计算不同 cutoff 下的相关统计量
        涉及统计量为：
            fpr: 假正率/特异度
            tpr: 真正率 / 灵敏度 / 召回率
            depth: 深度 / 响应率
            ppv: 精准率 / 命中率
            lift: 提升度
            afdr: 误判率
        """

        # 产生 cutoff 阈值
        threhold = self.threhold()
```

```
        # 产生指标列表
        fpr_list = []
        tpr_list = []
        depth_list = []
        ppv_list = []
        lift_list = []
        afdr_list = []

        # 遍历每一个 cutoff 下的指标，并输出结果
        for i in threhold:

            # cutoff 下产生 0，1 预测值
            pred = (self.pred > i).astype('int64')

            #产生指标
            fpr,tpr,depth,ppv,lift,afdr = self.confusion_matrix(pred = pred)

            #append 指标
            fpr_list.append(fpr)
            tpr_list.append(tpr)
            depth_list.append(depth)
            ppv_list.append(ppv)
            lift_list.append(lift)
            afdr_list.append(afdr)

        return(fpr_list,tpr_list,depth_list,ppv_list,lift_list,afdr_list)



def plot_roc(self,color='red',legend='model',title='Roc_Curve',xlabel='FPR',ylabel='TPR'):
        """绘制 ROC 曲线
        X 轴 : fpr
        Y 轴 : tpr
        参数 :
            color: 颜色
            legend: 图例名
            title: 标题
            xlabel: x 轴标签
            ylabel: y 轴标签
```

```
    """

    # 计算 auc
    auc = round(metrics._score(self.true,self.pred),2)

    # 计算 fpr(x 轴)和 tpr(y 轴)
    fpr,tpr,_,_,_,_ = self.index()

    #绘图
    plt.plot(fpr,tpr,
                color = color,
                label = legend + ' auc: {}'.format(auc),
                linestyle = '-')

    plt.plot([0,1],[0,1],
                color = 'black',
                linestyle = '--')

    plt.title(title)

    plt.xlabel(xlabel)
    plt.ylabel(ylabel)

    plt.legend(loc=4)


def plot_gains(self,color='red',legend='model',title='Gain_Curve',xlabel='Depth/RPP',ylabel='ppv'):
    """绘制增益曲线，该曲线常用于营销类模型
    X 轴 : depth
    Y 轴 : ppv
    参数 :
        color: 颜色
        legend: 图例名
        title: 标题
        xlabel: x 轴标签
        ylabel: y 轴标签
    """
```

```
    # 计算 depth(x 轴)和 ppv(y 轴)
    _,_,depth,ppv,_,_ = self.index()

    # 绘图
    plt.plot(depth,ppv,
                color = color,
                linestyle = '-',
                label = legend)

    plt.title(title)

    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.legend(loc=1)


def plot_lift(self,color='red',legend='model',title='Lift_Curve',xlabel='Depth/RPP',ylabel='Lift'):
    """绘制提升曲线，该曲线常用于营销类模型
    X 轴 : depth
    Y 轴 : lift
    参数 :
        color: 颜色
        legend: 图例名
        title: 标题
        xlabel: x 轴标签
        ylabel: y 轴标签
    """

    # 计算 depth(x 轴)和 lift(y 轴)
    _,_,depth,_,lift,_ = self.index()

    # 绘图
    plt.plot(depth,lift,
                color = color,
                linestyle = '-',
                label = legend)

    plt.title(title)
```

```
        plt.xlabel(xlabel)
        plt.ylabel(ylabel)
        plt.legend(loc=1)


    def plot_lorenz(self,color='red',legend='model',title='Lorenz_Curve',xlabel='Depth/RPP',ylabel='True Postive Rate'):
        """绘制洛伦兹曲线，该曲线常用于信用风险建模
        X 轴 : depth
        Y 轴 : tpr
        参数 :
            color: 颜色
            legend: 图例名
            title: 标题
            xlabel: x 轴标签
            ylabel: y 轴标签
        """

        # 计算 depth(x 轴)和 tpr(y 轴)
        _,tpr,depth,_,_,_ = self.index()

        #绘图
        plt.plot(depth,tpr,
                 color = color,
                 linestyle = '-',
                 label = legend)

        plt.plot([0,1],[0,1],
                 color = 'black',
                 linestyle = '--')

        plt.title(title)

        plt.xlabel(xlabel)
        plt.ylabel(ylabel)
        plt.legend(loc=4)
```

```
def plot_ks(self,color='red',legend='model',title='KS_Curve',xlabel='Depth/RPP',ylabel='KS statistic'):
    """绘制 KS 曲线，该曲线常用于信用风险建模
    X 轴 : depth
    Y 轴 : ks
    参数 :
        color: 颜色
        legend: 图例名
        title: 标题
        xlabel: x 轴标签
        ylabel: y 轴标签
    """

    # 计算 depth(x 轴)和 fpr,tpr(y 轴)
    fpr,tpr,depth,_,_,_ = self.index()
    ks = np.array(tpr) - np.array(fpr)
    ks_stats = round(max(ks),2) * 100

    #绘图
    plt.plot(depth,ks,
             color = color,
             linestyle = '-',
             label = legend + ' KS : {}'.format(ks_stats))

    plt.title(title)

    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.legend(loc=3)


def plot_afdr_adr(self,color='red',legend='model',xlabel='afdr',ylabel='recall',title='AFDR_ADR_Curve'):
    """绘制 afdr_adr 曲线,该曲线常用于欺诈侦测模型
    X 轴 : afdr
    Y 轴 : adr / tpr/RECALL
    参数 :
        color: 颜色
        legend: 图例名
```

```
            title: 标题
            xlabel: x 轴标签
            ylabel: y 轴标签
        """

        # 计算 afdr(x 轴)和 adr(y 轴)
        _,tpr,_,_,_,afdr = self.index()

        #绘图
        plt.plot(afdr,tpr,
                 color = color,
                 linestyle = '-',
                 label = legend)

        plt.title(title)

        plt.xlabel(xlabel)
        plt.ylabel(ylabel)
        plt.legend(loc=4)


    def
plot_precision_recall(self,color='red',legend='model',xlabel='recall',ylabel='precision',title='Precision_Recall_Curv
e'):
        """绘制 precision_recall 曲线,该曲线功能类似于 ROC，但样本不平衡时 ROC 曲线更佳稳健
        X 轴 : tpr/recall
        Y 轴 : ppv
        参数 :
            color: 颜色
            legend: 图例名
            title: 标题
            xlabel: x 轴标签
            ylabel: y 轴标签
        """

        # 计算 fpr(x 轴)和 ppv(y 轴)
        _,tpr,_,ppv,_,_ = self.index()
```

```
#绘图
plt.plot(tpr,ppv,
                color = color,
                linestyle = '-',
                label = legend)

plt.title(title)

plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.legend(loc=3)
```