“东证期货”全国大学生统计建模大赛

参 赛 论 文

题目：基于 XGBoost 与 Stacking 的

信用评分体系模型

队伍编号：738

参赛队员：梁思韵　上海财经大学

宋黄华　上海财经大学

赵　伟　上海财经大学

徐　博　复旦大学

凌　华　复旦大学

指导教师：骆司融

摘要

随着个人信用历史和信用行为数据的逐步完善，采用数据挖掘技术预测个人未来的信用表现日益成为主流方法。本文通过特征工程利用宏观因素和多家征信系统的数据构造新的有效特征，建立多种集成树以及与逻辑回归复合的模型构建个人信贷风险评估体系，并对个人信贷数据进行实证分析。结果表明，XGBoost 模型因其对高维连续性数据的适应性，以及通过对回归树残差建模，显著降低预测误差等优点，相对其他模型而言，具有更高的预测准确性和稳定性。在此基础上，本文针对不平衡样本的特点构造了 Ensemble-XGBoost 模型,并将预测概率转换为信用评分。更进一步，利用模型堆叠技术，通过多种基模型提取组合特征，使用逻辑回归模型进行汇总，使得模型效果再次提升。此外，本文更加注重模型的解释性，将 XGBoost 模型变成“白箱”系统，并使用 LIME 工具对模型做出的所有决策进行合理解释。

关键字：信贷风险评估，XGBoost，不平衡样本，Stacking， LIME

目录

图表目录

基于 XGBoost 与 Stacking 的信用评分体系模型

一. 绪论

金融是市场经济的核心部分，随着互联网信息化的深入发展，互联网金融凭借低门槛、普惠制的优势逐渐成为中国经济金融发展的创新力量。李克强总理在十二届全国人大二次会议上明确指出要促进互联网金融健康发展，完善金融监管协调机制，密切监测跨境资本流动，守住不发生系统性和区域性金融风险的底线，让金融成为一池活水，更好地浇灌小微企业、“三农”等实体经济之树。决策层的重视和肯定预示着互联网金融行业的春天已经到来。不少国内学者提出，以互联网为代表的现代信息科技，例如移动支付、云计算、社交网络等，将对人类金融模式产生根本影响，互联网金融模式在未来 20 年将成为主流。

目前全球经济、金融一体化的现实背景，加之互联网的蓬勃发展，出现越来越多的金融业务模式和运行机制，随之呈现的是白热化的竞争态势。金融机构能否抓住机遇，突破时间和地域的限制，为自身发展提供有力的风险保障，同时也为客户提供更好的服务体验。当下个人信贷业务将是我国金融业目前及未来发展的关键领域。而信用风险是金融业面临的最大风险。个人信用评估是个人信用制度的基础。合理的个人信用评估指标和科学的个人信用评估方法将大大推动信贷机构个人信用业务的开展。当前国内个人信用体系尚不健全，互联网金融行业无法完全利用银行间的征信系统，风险管理难度进一步提高。有关数据表明，高质量的信用评估可以大大简化手续、提高评估效率和准确度，进一步的可以在减轻机构信贷管理成本的基础上，提高客户满意度和信贷业务效益。因而个人信用评估对于信贷业务的发展和社会信用体系的完善具有重要意义，个人信用评估模型的探索有助于运营者优化金融风险控制手段、保护投资者和自身的合法权益，有助于金融借贷机构的健康、长远发展，有助于市场和政府调整信贷结构、实现社会资源的合理配置。

信用风险评分模型是银行等信贷发放机构应用较为成熟的风险计量工具。早在 20 世纪 40 年代，美国有些银行就开始尝试性研究信用评分方法，用于快速处理大量信贷申请。1956 年，工程师 Bill Fair 和数学家 Earl Isaac 共同发明了著名的 FICO 评分方法。该方法基本以 Logistic 回归方法为技术核心，是当前业界应用最成熟的信用风险评分模型。目前征信机构通过采集海量的个人信用历史和信用行为数据，采用数据挖掘方法得出的信用行为模式能够更加准确地预测个人未来的信用表现，能够提高操作的效率，降低授信成本，精确估计消费信贷的风

险，是金融机构内部评分不可替代的重要工具。

本文主要通过建立信用评分模型,如何在个人信贷业务中嵌入信用评分模型。

二．模型理论基础

近年来许多模型被应用在信用评分系统上，诸如支持向量机，决策树，聚类分析，朴素贝叶斯，LDA，QDA，简单神经网络等。在以上的模型中，简单神经网络通常具有更好的效果，但是作为黑箱系统，它的可解释性大大降低。支持向量机可以克服神经网络的不足，包括小样本下泛化能力不足和结构选择的问题。但是，对于高维复杂数据时，这些模型存在不能主动进行特征选择和特征组合的问题，因此准确率会受到无关维度的影响。并且，异常点也会对分类结果产生十分严重的影响。对于 Logistic 回归模型，特征组合非常关键，但只能通过人工经验，耗时耗力，并且效果不一定得到提升。

当前国内银行等信贷系统的主流信用评估模型主要是参考美国 FICO 的评分体系，其核心是 Logistic 回归算法。该算法的特点在于简单易部署，得到的结果为概率值，可以人为确定 cutoff，训练速度快，算法可分布。最重要的在于 Logistic 回归模型的结果直观，便于理解，可以与人的直观感受相结合评估模型效果。但是不足之处在于：Logistic 模型常用于线性问题，对于非线性问题需要定义非线性函数对特征进行映射，如果特征出现多重共线性的问题，会降低预测准确率；另外，如果特征的解释性不足，特别是当数据量较大，而特征变量较少的时候，采用 Logistic 回归模型往往得不到理想的效果。因此 Logistic 模型要取得较好的效果，必须在特征工程上花费巨大的时间和精力。考虑到我国基本国情以及互联网信用数据的特点,笔者认为一些基于集成树的机器学习算法,比如 GBDT, XGBoost,更为适合。

Facebook 在 2014 年提出使用 GBDT 和 Logistic 回归的融合模型应用在广告点击率（CTR）的预测上，使用 GBDT 模型提取组合特征，再用 Logistic 回归模型预测,成功将 AUC 提高了三个百分点。而本文在比较了决策树、随机森林、GBDT、XGBoost 以及多种融合模型的效果后，决定将 XGBoost 与 Logistic 回归的融合模型运用到信用评分系统中去。因此，笔者首先对后文涉及的算法和模型进行如下简单介绍。

(一)集成树模型

集成学习，顾名思义是通过构建并结合集成多个学习器来完成学习任务，有

时也被称为多分类系统。集成学习要获得好的结果应做到“好而不同”，即个体学习器要有一定的准确性，并且学习器之间应该有差异。

集成学习器优于单一学习器的原因有以下三点：首先，对于单一的学习器而言，训练数据并不能提供足够的信息。比如在同一个训练数据上存在很多学习器的表现同样好，因此将这些学习器组合起来可以进一步提升效果。其次，单一学习器寻找最优解的算法可能并不完美，即使存在全局最优解，算法也可能收敛到局部最优解上，因此集成学习器可以对此进行补充。最后，最优解虽然可能达不到，但是集成学习器可以用近似解去逼近真实解。

集成树模型是以决策树为基础，通过构建并结合集成多个学习器来完成学习任务的。集成树模型可以根据个体学习器之间的关系分为 Boosting 和 Bagging，前者个体学习器之间存在强依赖关系、必须串行生成；后者个体学习器之间不存在强依赖关系、可同时并行生成。Boosting 是一族将弱学习器提升为强学习器的算法，其工作机制如下：先从初始训练集中训练出一个基学习器，再根据基学习器的表现对训练样本进行调整，使得先前基学习器做的训练样本在后继受到更多的关注，然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器的数目达到实现指定的值 T，最终将这 T 个学习器进行加权结合。而 Bagging 是并行式集成学习方法的代表，使用自助采样（bootstrap sampling）即有放回抽样，训练多个学习器。在对训练输出进行结合时，Bagging 通常对分类器使用简单的投票法，对回归任务使用简单的平均法。

从 bias-variance tradeoff 的角度讲，Boosting 通过对残差建模降低模型的 bias（偏差），Bagging 通过多个独立的弱学习器投票，降低模型的 variance（方差）。因此前者能基于泛化性能相当弱的学习器构建出很强的集成，而后者在不剪枝决策树，神经网络等易受样本扰动的学习器上效果更显著。

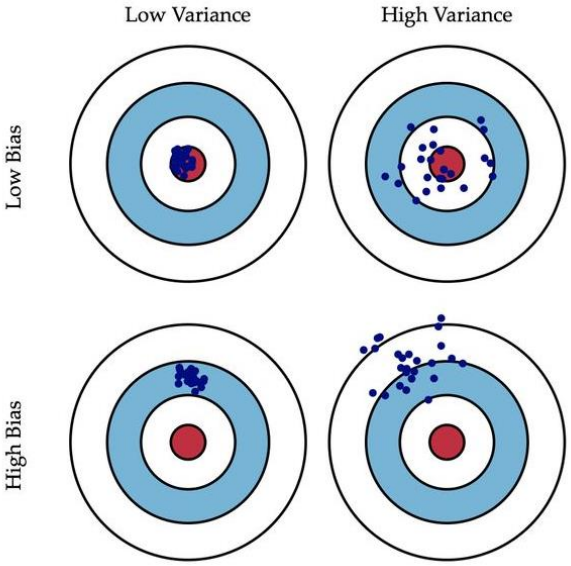

图 2.1 bias-variance tradeoff

(二)XGBoost 算法

XGBoost(eXtreme Gradient Boosting),是一种基于梯度提升算法(Gradient Boosting）以及决策树（DecisionTree）的改进型学习算法。其原理是使用迭代运算的思想，将大量的弱分类器转化成强分类器，以实现准确的分类效果。XGBoost 算法由华盛顿大学陈天奇博士提出，由于其性能的高效，近几年来被越来越多的应用于机器学习领域。

XGBoost 是 Boosting 中的经典方法，它与决策树是息息相关的，它通过将很多的决策树集成起来，从而得到一个很强的分类器。其中决策树就是一种对空间不断进行划分算法，通过给每个划分的空间赋予一个标签或权重，那么当样本落到这个空间里面，就认为这个样本就满足这个标签。boosting 的核心思想就是希望训练出 K 颗树，将它们集成起来从而预测 Y。

XGBoost 算法的原理是将原始数据集分割成多个子数据集,将每个子数据集随机的分配给基分类器进行预测,然后将弱分类的结果按照一定的权重进行计算,一次来预测最后的结果。更通俗的讲，提升算法的原理就像接力赛跑，每一棒的选手都是从很多跑的没那么快的选手中挑选出来的,而这些挑选出来的选手一般来说都是跑的最快的，最后一起赢得比赛。基分类器只有一起协作才能取得良好的效果，但传统的 SGD 模型并不是这样的，常规的 SGD 算法的目的是找到一个能够使平方损失函数最小的函数。一般将这种迭加基分类器结果的预测模型称为加性模型。公式如下：

$$\mathrm{F} = \sum_{i=1}^{m-1} f + f_m \qquad (1)$$

加性模型的预测拟合是对每个基分类器的预测结果进行向后拟合算法的样条平滑，使得拟合误差即偏度和自由度即方差之间达到均衡状态。本文基函数选择的是回归树，有的文献也将其称为 CART 树，选择回归树是立足于我们原有的经验。当树的深度足够深时，其作为分类器的效果会非常的好。因此，选择 CART 树作为模型的基函数，那么单个 CART 第 M 次预测的结果为：

$$f_m(x) = T(X;\theta_m) \qquad (2)$$

由此，基函数已经确定，其中 T 为决策树，m 代表基分类器的数量，theta 代表决策树的划分路径,最后的预测结果为前一次的预测结果加上当下的决策树,而误差项可以表示为：

$$\mathrm{L}(y,\hat{y})=\mathrm{L}(y,f_{m-1}(x))+\mathrm{T}(X,\theta_m) \qquad (3)$$

这里 L(y, $\hat{y}_i$)是真实值y_i和预测值$\hat{y}_i$之间差值之和。至此，我们的误差损失已经量化完成。

基尼系数、剪枝和控制树的深度是 CART 进行分类的重要手段。实际是通过

上述方式控制模型的方差和偏差，使得模型的拟合泛化能力更强。本文也是基于这种原理来设计函数。例如，可以用叶子节点数目 T 和 Leaf Score 的 L2 模的平方来定义结构复杂度函数：

$$\emptyset(\theta) = \gamma T + \frac{1}{2}\ \lambda\ \sum_{J=1}^{T}\|W\|_J^2 \tag{4}$$

其中γ表示控制树复杂度的系数，相当于给 XGBoost 算法模型的树做了前剪枝，而 λ 表示通过多大的比例来改变正则项，相当于给复杂的模型一个惩罚，防止模型出现过拟合。

综合偏差函数和方差函数可以给出以下目标函数：

$$Obj(\theta)=\sum_i l(y_i + \hat{y}_i) + \gamma T + \frac{1}{2}\ \lambda\ \sum_{J=1}^{T}\|W\|_J^2 \tag{5}$$

前人对于 Gradient Boosting Descent Tree 算法的处理方法是反复计算目标函数(5)的误差，使其误差越来越小，这种方法即为梯度下降算法。只要每次都按照这种方式得出弱分类器，再将每个弱分类器进行相加，最后的模型得出的结果必然是最优的。梯度下降的公式如下所示：

$$-\left[\frac{\partial L(y,f(x_i))}{\partial f(x_i)}\right] f(x) = f_{m-1}(x) \tag{6}$$

按照公式(6)对损失函数求导之后，树的个子节点和每个节点的权重都可以确定，所以树也就确定了。但是，由于 XGBoost 算法所使用的基分类器数量较多，因此，我们需要更为通用的算法来实现梯度下降，XGBoost 算法的发明者使用的是泰勒二阶展开替代了原来的一阶导数，使得算法更具普遍性。加入泰勒二阶展开之后的目标函数如下：

$$Obj(\theta)=\sum_{i=1}^{n} l(y_i, \hat{y}_i^{m-1} + f_m(x_i)) + \emptyset(f_m) \tag{7}$$

式中，n 表示所用的样本数量，m 表示当前迭代的次数，f(m)表示当前迭代的误差。

用泰勒展开：

$$F(x+\Delta x) \cong f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2 \tag{8}$$

定义：

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{m-1})}{\partial \hat{y}_i^{m-1}} \qquad h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{m-1})}{\partial \hat{y}_i^{m-1}} \tag{9}$$

代入计算：

$$Obj_m \cong \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}_i^{m-1}\right) + g_i f_m(x_i) + \frac{1}{2}h_i f_m^2(x_i)\right] + \emptyset(f_m) \tag{10}$$

为了清楚的描述 XGBoost 的算法过程，这里用以下伪代码表

Input： X：训练集，且$|X| = n$

y：样本标记

l：损失函数

f：基分类器

output：集成分类器 F(X)=$F_K(X)$

for k←1 **to** K **do**

for i←1 **to n do**

$g_i \leftarrow \frac{\partial l(y_i, \hat{y_i}^{m-1})}{\partial \hat{y_i}^{m-1}}$

$h_i \leftarrow \frac{\partial^2 l(y_i, \hat{y_i}^{m-1})}{\partial \hat{y_i}^{m-1}}$

end

使用这些一阶和二阶梯度优化目标函数$\mathrm{Obj}^{(k)}$；

f_k ←贪心地生成一棵树；

$F_k \leftarrow F_{k-1}(X)+\varepsilon f_k(X)$；

end

注：其中，$\hat{y}_i^k$表示样本x_i在第 k 轮的预测值。

图 2.2　XGBoost 的算法过程伪代码

(三)集成树与逻辑回归融合模型

该模型的原理在于将原始数据通过集成树模型（如 XGBoost 模型）生成新的组合特征，再将新的特征与原始的特征组成新的特征变量训练 Logistic 模型。如下图 2.3 所示，原始数据进入集成树模型得到的两棵决策树中，决策树每一个叶子代表一个新的特征，若样本落在某一个叶子中则该特征设置为 1，否则设置为 0。图中包含决策树 Tree1 和 Tree2 共五个叶子，假设某个样本经过 Tree1 落在第一个叶子中，经过 Tree2 落在第二个叶子中，则构造的新的特征向量是[1,0,0,0,1]。

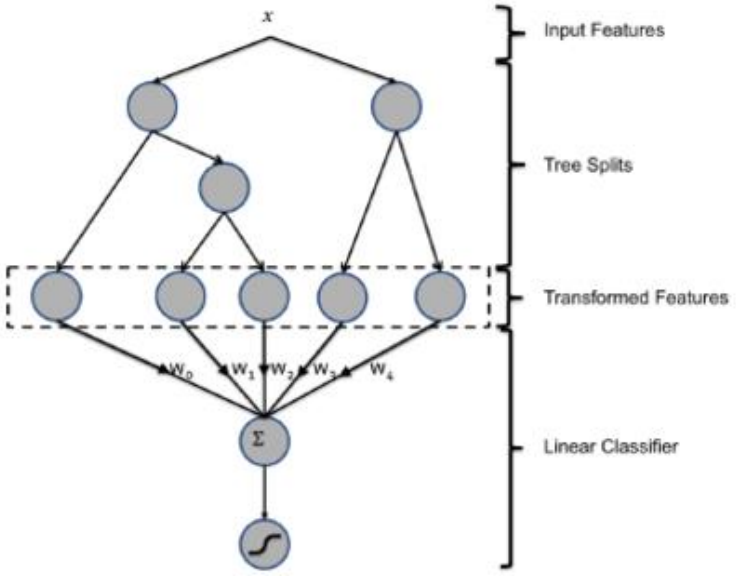

图 2.3 集成树模型构造特征示意图

(四)Local Interpretable Model-Agnostic Explanations

集成树模型相比较于传统 Logistic 回归模型，虽然在准确率和稳定性上都有显著的提升，但是复杂模型如同一个黑盒，几乎无法感知它内部的工作过程，解释性大打折扣。直观上看，解释每次独立预测背后的基本原理能使我们更容易信任或是不信任预测结果，乃至分类器本身。即使我们无法理解模型在所有情况下的表现，却有可能理解（大多数情况都是如此）它在某些特定情况下的行为。

模型的可解释性在实际应用中有着举足轻重的重要性，在量化分析过程中，模型的解释能力甚至超过模型本身的预测能力。比如，信贷部门根据模型输出的结果拒绝某客户的贷款申请，但是如果不给出具体的拒绝理由可能会造成一定的法律纠纷。所以，成熟的信贷模型应该能够给出拒绝或接受业务的理由，诸如给出违约记录、月收入等因素是模型作出该决策的主要因由。

本文使用 Local Interpretable Model-Agnostic Explanations（以下简称 LIME）模型能对以上集成树模型的结果做出一定程度的解释。该模型能够局部保真，真实地反映分类器在预测样本上的行为，并且这个解释是可以理解的。LIME 能够解释任何模型，而不需要进行模型适配，所以是与模型无关的。

LIME 的原理是在考察预测样本的周围某个小范围内分类器模型的预测行为，然后根据这些扰动的数据点距离原始数据的距离分配权重，基于它们学习到一个可解释的模型和预测结果。直观的说就是在预测样本周围的小范围内，拟合一个线性分类器去近似替代原非线性分类器。如下图 2.4 所示，原始模型的决策函数

用蓝/粉背景表示，显然是非线性的。加粗红色的“+”表示被解释的样本（记为X)。我们在X周围采样，按照它们到X的距离赋予权重(这里权重的意思是尺寸)。我们用原始模型预测这些扰动过的样本，然后学习一个线性模型（虚线）在 X 附近很好地近似模型。注意，这个解释只在 X 附近成立，对全局无效。

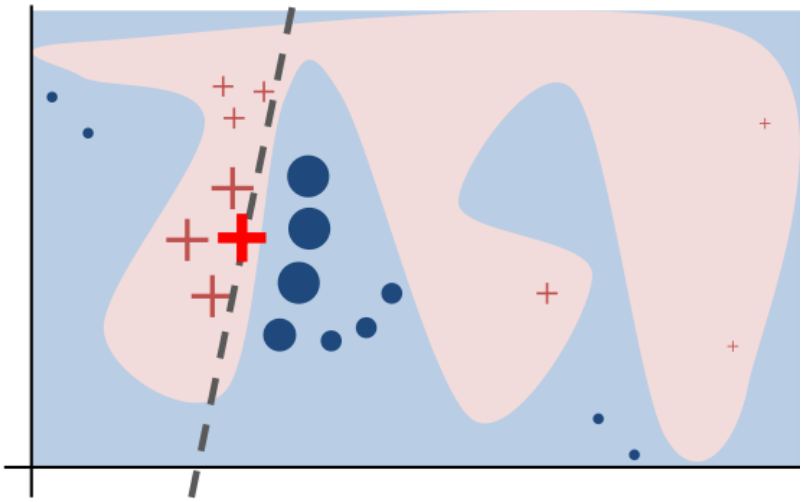

图 2.4 LIME 原理

(五)stacking 模型堆叠技术

Stacking 是模型融合的方法之一，它能够将一系列模型（基模型）的输出结果作为新特征输入到其他模型中去，这种方法实现了模型的堆叠，即第一层的模型输出作为第二层的模型输入，第二层的模型输出作为第三层的模型输入，依次类推，最后一层模型输出结果作为最终结果。Stacking 的思想类似于卷积神经网络(CNN)，认为不同基模型就是不同的卷积核，它们从数据的不同角度出发，提取到的特征各有差异和优缺点，Stacking 利用第二次模型，通常为 Logistic 回归模型，将所有基模型的优点统一起来，以提升整体模型的效果。但是，基模型之间的相关性要尽可能的小，从而能够互补模型间的优势；基模型之间的性能表现不能差距太大，太差的模型会拉低整体的水平。

为了避免用相同的数据训练和测试模型而过拟合现象，实际中会引入 K-Fold 交叉验证（cross validation)。对于一个基模型（如 XGBoost 模型）我们有训练样本和验证样本：

（1）将训练数据随机打乱并 K 等分，每次取其中一份作为测试集，(K-1）份作为训练集，用训练集训练模型，并预测测试集和验证样本。

（2）如此重复 K 次，将 K 个训练集预测的结果拼接起来作为该基模型的输出，K 个验证样本的预测结果取平均作为该基模型的特征。

（3）以上步骤在 M 个不同的基模型上重复，得到 M 个特征的训练样本作为第二层模型的输入训练，并预测验证样本。

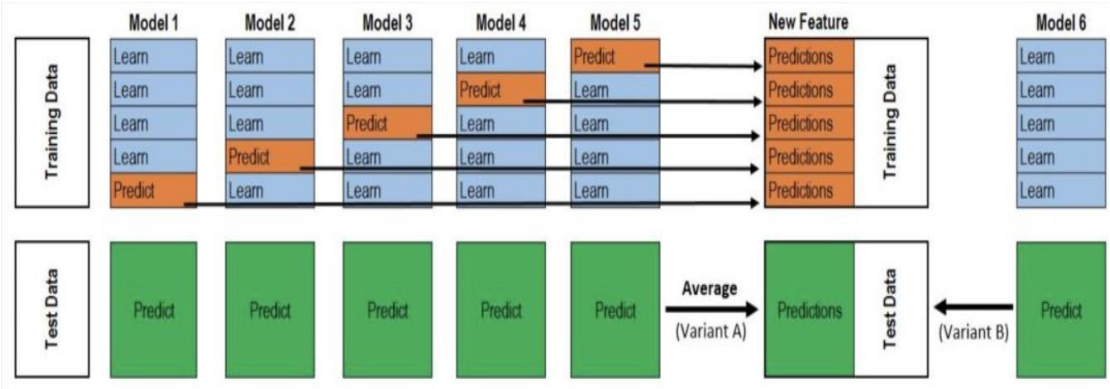

图 2.5　Stacking 原理

三．　数据获取及处理

（一）原始数据集的预处理和划分

本文共获得了 30000 个借贷者的个人信息及信用记录，在去除与因变量无直接关系或直接关系较小的变量以及恒为常数的变量后将剩余变量汇总如表 3.1。

首先，对于数据中的异常值及缺失值进行分析处理。对于数值型变量，数值型变量描述性统计量及缺失概率汇总见表 3.2，可以看出各变量无明显异常值，且缺失值所占比例均较小，因此暂时不进行进一步处理。对于分类型变量，每个变量中缺失值比例也均较少，因此暂时不进行处理（各分类变量的频数频率表见附件）。

表 3.1　描述性统计分析变量汇总

表名	字段名	中文解释	变量类型
基础表-训练集	Agent	客户渠道	分类
	is_local	是否本地籍	分类
	work_province	工作城市	分类
	education	教育	分类
	has_fund	是否有公积金	分类
	marriage	婚姻	分类
	salary	收入	分类
	Y	是否逾期(目标变量值)	数值
机构版征信-报告主表	query_reason	查询原因	分类
	query_org	查询机构	分类
机构版征信-信用提示	house_loan_count	个人住房贷款笔数	数值
	commercial_loan_count	个人商用房(包括商住两用)贷款笔数	数值
	other_loan_count	其他贷款笔数	数值
	loancard_count	贷记卡账户数	数值
	standard_loancard_count	准贷记卡账户数	数值

表 3.2　数值型变量描述性统计量

变量标签	非缺失个数	最小值	最大值	均值	标准差	缺失概率	缺失概率<=0.98
个人住房贷款笔数	29971	0	7	0.13813	0.3891	0.00097	1
个人商用房(包括商住两用)贷款笔数	29971	0	3	0.00394	0.0706	0.00097	1
其他贷款笔数	29971	0	342	8.01845	12.3597	0.00097	1
贷记卡账户数	29971	0	116	7.97171	6.2585	0.00097	1
准贷记卡账户数	29971	0	12	0.15652	0.5450	0.00097	1

对于工作省份这一变量，由于工作省份分类较多，因此本文根据各个省份的 WOE(Weight of Evidence)值进行聚类，得到工作省份的四个新的分类，分别为：Missing、<250000、250000-350000、>350000。通过查阅我国地址码编制规则，可以得到<250000 为东北和华北地区的省份，250000-350000 为华东地区的省份、>350000 其他地区的省份。进一步印证了聚类的合理性。新的分类 WOE 值见图 3.2，可以看出，重新分类后各类别 WOE 值差异明显。

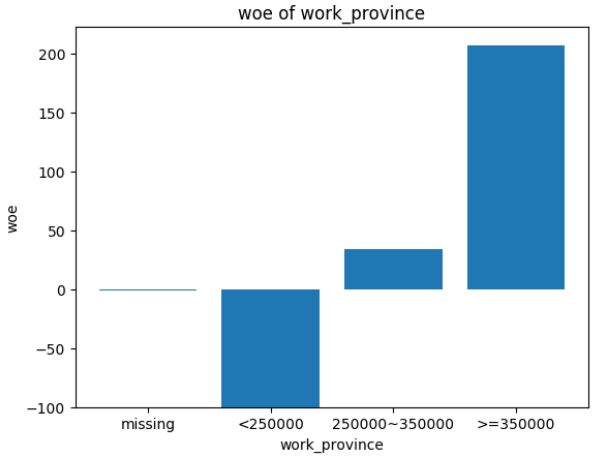

图 3.2 工作省份聚类结果的 WOE 值

其次，研究自变量间是否存在共线性问题。由表 3.5 可以看出，各变量间相关系数均低于 0.3，具有较低的相关性。

表 3.5　数值型自变量间相关系数

变量名	个人住房贷款笔数	个人商用房（包括商住两用）贷款笔数	其他贷款笔数	贷记卡账户数	准贷记卡账户数
个人住房贷款笔数	1	0.02027	0.07552	0.14971	0.08008
		0.0004	<.0001	<.0001	<.0001
个人商用房(包括商住两用) 贷款笔数	0.02027	1	0.02334	0.03452	0.01346
	0.0004		<.0001	<.0001	0.0198
其他贷款笔数	0.07552	0.02334	1	0.21293	0.05955
	<.0001	<.0001		<.0001	<.0001
贷记卡账户数	0.14971	0.03452	0.21293	1	0.2032
	<.0001	<.0001	<.0001		<.0001
准贷记卡账户数	0.08008	0.01346	0.05955	0.2032	1
	<.0001	0.0198	<.0001	<.0001	

最后，由于分类器模型中，诸多基于距离度量的分类器对于特征的单位非常敏感，数据间数量级差别较大会造成其分类误差较大，为了消除数据数量级差别的影响，使对比实验更有说服力，实验在进行模型训练前对特征进行标准化。本文中使用的标准化方法为最小-最大规范化，即：

$$x_i' = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{11}$$

本文应用的数据包含 30000 个样本，贷款时间为 2017 年 1 月 4 日至 2017 年 6 月 30 日。测试数据包含 10000 个样本，贷款时间为 2017 年 7 月 1 日至 2017 年 9 月 30 日。从贷款时间上的分布可以看出，模型需要根据历史数据预测未来客户的违约概率。因此，我们提取样本中贷款时间靠后（2017 年 6 月 27 日-2017 年 6 月 30 日）的 2000 个样本作为验证样本，剩余的 28000 个样本作为训练样本，用于训练和比较不同的模型。

（二）描述性统计分析

为了更好的对数据进行整体把握，为后文特征工程与模型建立提供参考，本文利用描述性统计分析的方法，从个人基本信息与个人信用记录两个方面分别对各变量与信贷违约关系进行初步理解与分析。

另外，由于本文信贷记录为不平衡数据，未发生逾期观测占绝对多数，占比高达 93.75%。因此为消除同一变量不同类别人数可能不同的影响，本文对于分类变量与目标变量关系进行分析时使用列（或行）轮廓。

1）个人基本信息与信贷违约描述性统计分析

通过对个人基本信息与信贷违约描述性统计分析发现，借贷者收入、借款渠道、学历、婚姻状况和户籍五个方面的个人信息对其还款能力影响较大。

在收入方面，收入越高的客户经济实力越强，往往信贷逾期概率越小，如图 3.3。 客户借款渠道与预期情况间存在较强相关性，如图 3.4。主要原因可能是各渠道的用户群体情况存在差异。学历也会对还款情况造成影响，如图 3.5，学历越高逾期概率越小，硕士及以上学历者逾期率仅为 2.5%，需要指出的是学历项为缺失的用户逾期可能性最高。就婚姻状况而言，离婚的群体逾期率最高；就是否是本地籍而言，本地籍群体逾期率较高。

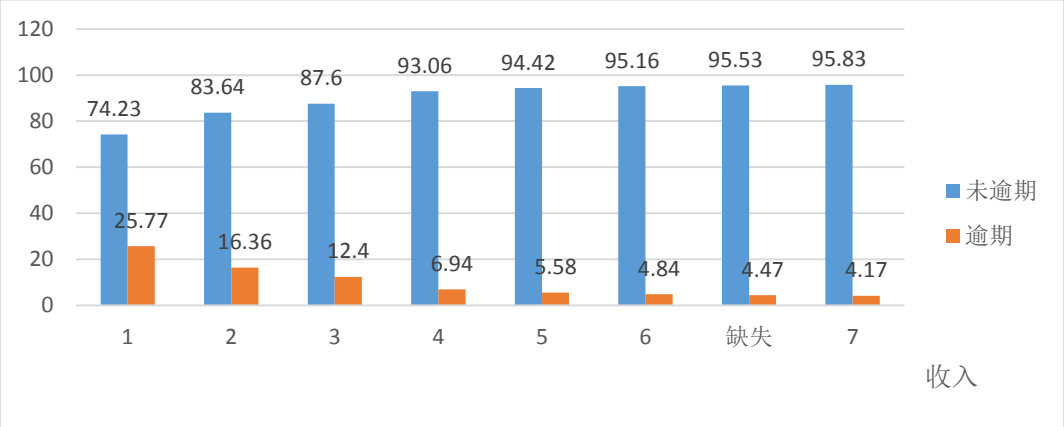

图 3.3 收入与逾期比例条形图

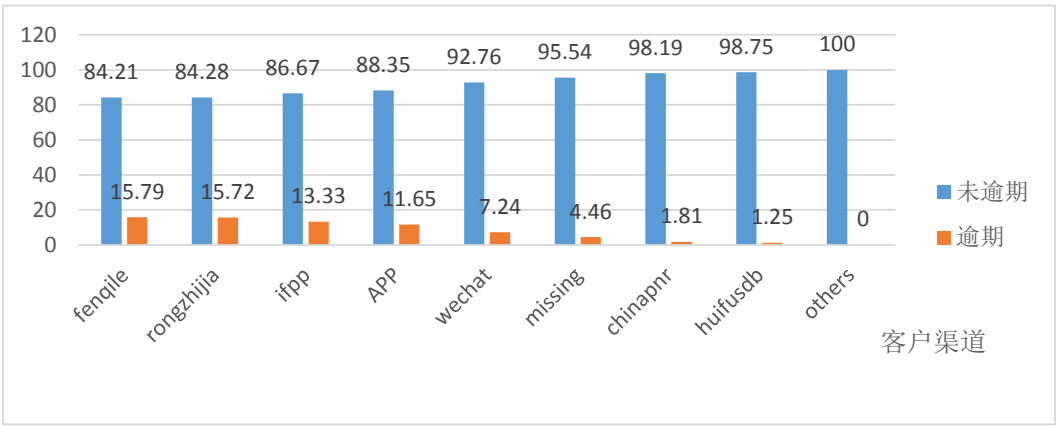

图 3.4 客户渠道与逾期比例条形图

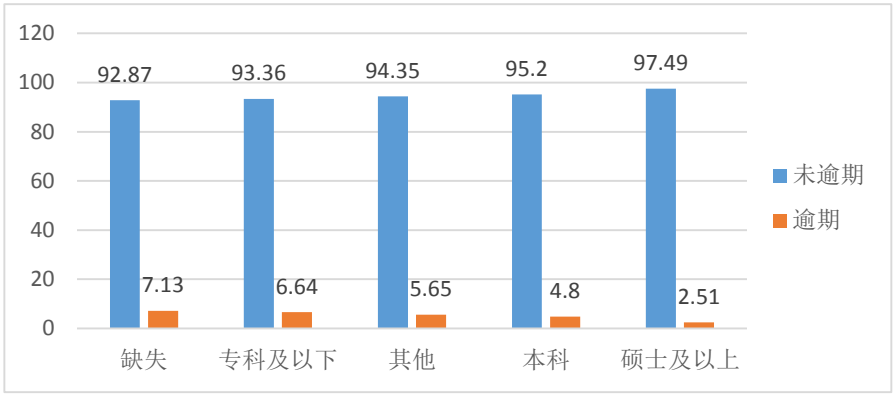

图 3.5 学历与逾期比例条形图

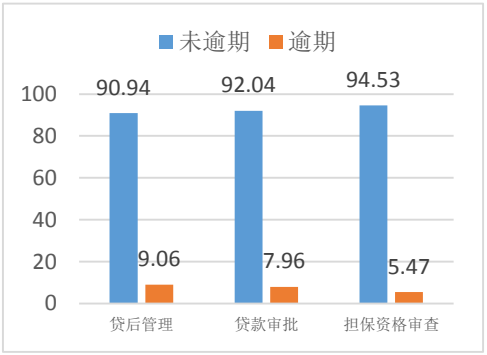

图 3.6 查询原因与逾期比例条形图

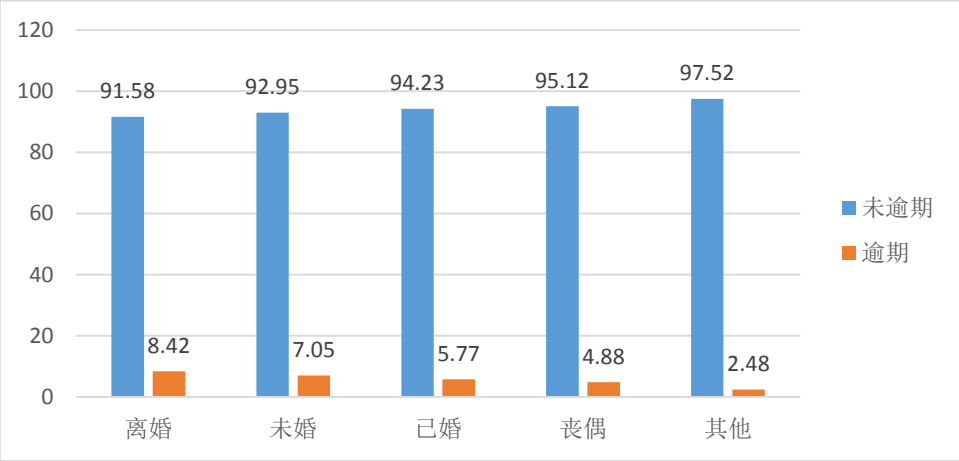

图 3.7 婚姻状况与逾期比例条形图

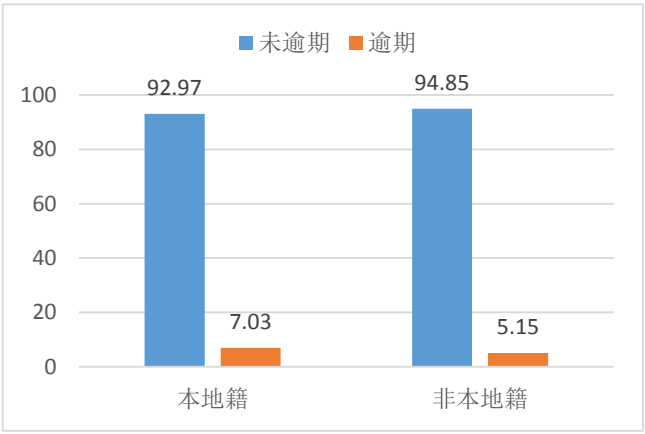

图 3.8 是否本地籍与逾期比例条形图

2）个人信用记录与信贷违约描述性统计分析

除个人基本信息外，借贷者的借贷查询记录、贷款笔数等个人信用记录也会对信贷违约情况造成影响。

如图 3.6，就查询原因而言，贷后管理逾期率最大，担保资格审查逾期率最小，这是因为在贷后进行信用查询往往意味着贷款机构可能获得的某些信息显示这笔贷款逾期可能性较大或者已经逾期，因此为了进行贷后管理与追踪该机构再次查询了该客户的信用信息。而进行担保资格审查逾期率最小则是因为贷款担保人往往会具有良好的信用情况。

计算出借贷者细分用途的贷款笔数与违约情况的 speraman 和 hoeffding 两种相关系数，并对这两种相关系数排序后作图 3.1，可见两种相关系数计算结果均显示，与违约情况相关性最高的是贷记卡账户数且方向为负，这可能是因为拥有贷记卡较多的用户通常信用状况良好。最不相关的是个人商用房（包括商住两用）贷款笔数且也成负相关，这可能是因为申请个人商用住房贷款的人数较少，所以该变量中包含的关于个人信用信息较少。另外，总体来看各自变量与目标变

量相关系数均较低。

表 3.6　数值型自变量与因变量间 speraman 和 hoeffding 相关系数

变量名	Spearman correlation	Spearman p-value	Hoeffding corelation	Hoeffding p-value
贷记卡账户数	-0.090355	0	0.000330326	0.00001
其他贷款笔数	-0.061343	0	0.000120566	0.00038
个人住房贷款笔数	-0.029004	0	-0.000042063	1
准贷记卡账户数	-0.028803	0	-0.000044507	1
个人商用房（包括商住两用）贷款笔数	-0.005987	0.30003	-0.000058679	1

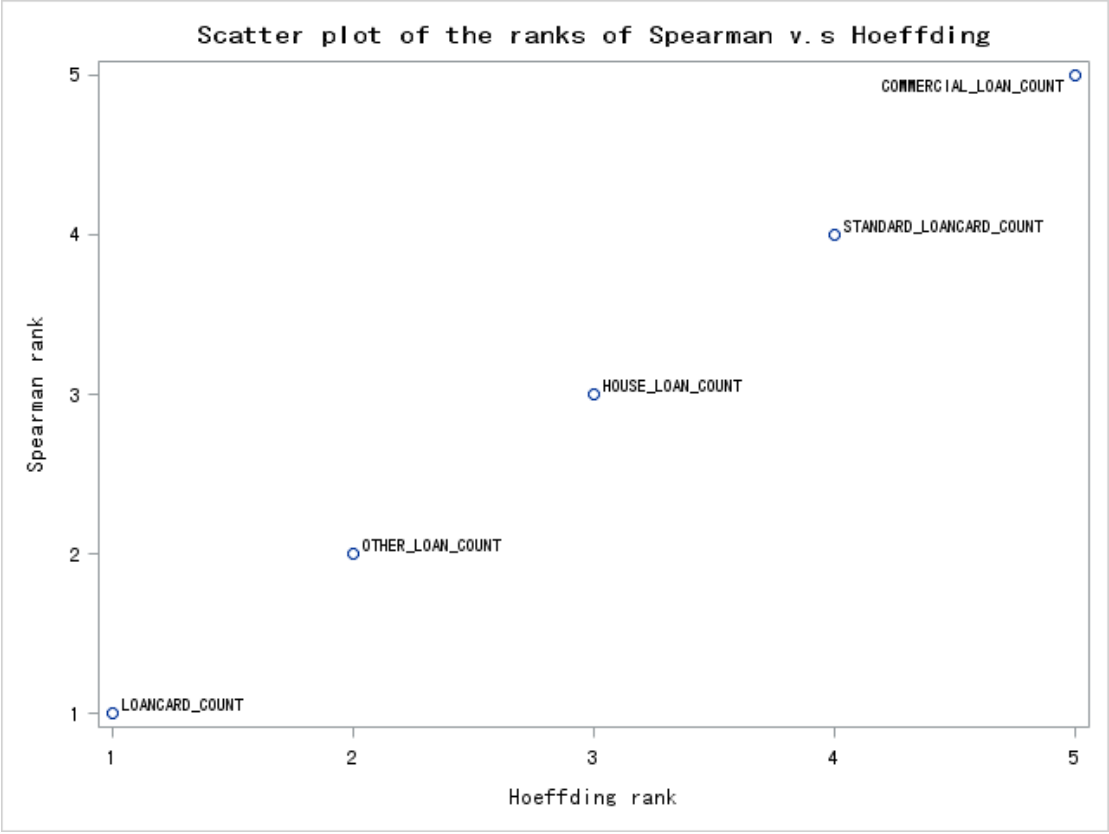

图 3.9 数值型自变量与因变量间 speraman 和 hoeffding 相关系数顺序图

(三)特征工程

为了最大限度地从原始数据中提取特征以供算法和模型使用，进而做到较优的预测，我们采用特征工程提取变量。首先，根据可用性对特征使用方案进行评估，我们删去了一些对因变量影响较小的自变量，例如贷款机构、发放日期等。第二，某些机器学习算法和模型只能接受定量特征的输入，那么需要将定性特征转换为定量特征。最简单的方式是为每一种定性值指定一个定量值，但是这种方式过于灵活，增加了调参的工作。我们采用哑编码的方式将定性特征转换为定量特征：假设有 N 种定性值，则将这一个特征扩展为 N 种特征，当原始特征值为第 i 种定性值时，第 i 个扩展特征赋值为 1，其他扩展特征赋值为 0。哑编码的方式相比直接指定的方式，不用增加调参的工作，对于线性模型来说，使用哑编码后的特征可达到非线性的效果。第三，对于每一个订单都包含多条的数据，例如用户在不同机构贷款的金额值，我们将每个订单的数据加和得到总值作为新的变量。最后保留的变量如表 3.7.

当数据预处理完成后，我们需要选择有意义的特征输入机器学习的算法和模型进行训练。通常来说，从两个方面考虑来选择特征：特征是否发散：如果一个特征不发散，例如方差接近于 0，也就是说样本在这个特征上基本上没有差异，这个特征对于样本的区分并没有什么用；特征与目标的相关性：与目标相关性高的特征，应当优选选择。根据特征选择的形式又可以将特征选择方法分为 3 种：Filter 过滤法，按照发散性或者相关性对各个特征进行评分，设定阈值或者待选择阈值的个数，选择特征；Wrapper 包装法，根据目标函数（通常是预测效果评分），每次选择若干特征，或者排除若干特征；Embedded 集成法，先使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征。类似于 Filter 方法，但是是通过训练来确定特征的优劣。

表 3.7 最终变量汇总表

逾期	curr_overdue_cyc_max	信贷账户当前最大逾期期数（贷款）
	curr_overdue_cyc_card_max	信贷账户当前最大逾期期数（贷记卡）
	curr_overdue_amount_card_sum	信贷账户当前逾期总金额（贷记卡）
	overdue_amount_sum	信贷账户当前逾期总金额
	normal_loan_count	信贷账户当前逾期总金额
	normal_loan_card_count	当前状态正常的信贷账户数量（贷款）
	normal_count	当前状态正常的信贷账户总数量
	his_count_dw	征信记录上信贷账户的逾期总笔数（贷款）
	his_overdue_cyc	征信记录上信贷账户的最大逾期期数（贷款）
	his_max_duration	征信记录上信贷账户的单月最高逾期总额（贷款）
	hi_oa_per_mon	征信记录上信贷账户的最大贷款时长（贷款）
	his_count_dw_card	征信记录上信贷账户的逾期总笔数（贷记卡）
	his_overdue_cyc_card	征信记录上信贷账户的最大逾期期数（贷记卡）
	his_amount_card	征信记录上信贷账户的逾期总金额（贷记卡）
	his_count	征信记录上信贷账户的逾期总笔数
	last24_overdue_count	征信记录上过去 24 个月有过 1 个月及以上逾期的账户数（贷款）
	last24_noverdue_count	征信记录上过去 24 个月从未有过逾期的账户数（贷款）
	last24_od_cyc_max	征信记录上过去 24 个月最大逾期期数（贷款）
	near_overdue_month_card	征信记录上最近一次 1 个月及以上逾期距查询日期的月份数（贷记卡）
	last24_noverdue_count_card	征信记录上过去 24 个月从未有过逾期的账户数（贷记卡）
	min_near_overdue_month	征信记录上最近一次 1 个月及以上逾期距查询日期的月份数
	last24_noverdue_count_sum	征信记录上过去 24 个月从未有过逾期的账户数
个人基本信息	Y	目标变量值
	ID_CARD	身份证
	LOAN_DATE	放款时间
	AGENT	客户渠道
	IS_LOCAL	是否本地籍
	WORK_PROVINCE	工作城市
	EDU_LEVEL	教育水平
	MARRY_STATUS	婚姻状态
	SALARY	收入
	HAS_FUND	是否有公积金
账户账	daymax	信贷账户最大账龄（单位天数）
	daymin	信贷账户最小账龄（单位天数）
	COUNT	信贷账户个数

龄	sumbalcre	信贷账户本金余额
授信余额	totalcre	信贷账户授信总额
	sumsch	信贷账户当前应还金额
	sumactpay	信贷账户实际还款金额
	balrate	信贷账户额度使用率
	account	贷款账户个数
	crelim	贷款账户授信总额

(四)宏观因素

宏观环境对于贷款的违约率是有一定影响的，经济上行阶段的违约率较低，经济下行阶段违约率较低。我们观察训练数据在不同时间上的违约率分布情况可以发现，其呈现明显的波浪形状，与四阶段经济周期曲线相似。基于美林投资时钟理论《The investment clock, special report: Making money from macro》，使用经济增长率与通胀率指标对经济周期进行划分，本文使用工业增加值的月度同比数据代表经济增长，CPI 月度同比增长代表通胀率，将宏观环境划分为衰退、复苏、过热和滞涨四个阶段。对于部分时间经济阶段切换过快的现象，本文采用滤波进行降噪处理，可以看出 2004-2010 年经济周期的轮换比较明显，近年来切换比较频繁。

对于宏观变量的使用，我们有两种不同的角度。通常可以将宏观变量作为哑变量解释样本的违约情况，但是基于测试数据时间（三个月）较短，绝大多数样本所处的经济阶段相同，使用该变量的效果不显著。因此，本文采取的方法是，把宏观变量结合进阈值的选择，经济环境上行的时候提高阈值，降低预测为可能违约客户的比例；反之，经济下行的时候降低阈值。

表 3.8 四个经济阶段划分

阶段	增长率	通胀率
衰退	下降	下降
复苏	上升	下降
过热	上升	上升
滞涨	下降	上升

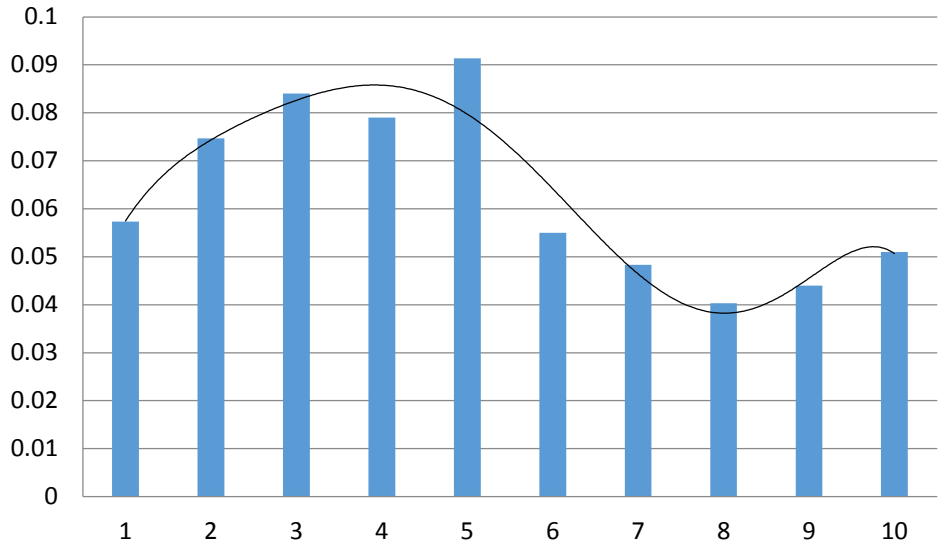

图 3.10　2017 年 1 月-6 月贷款违约率

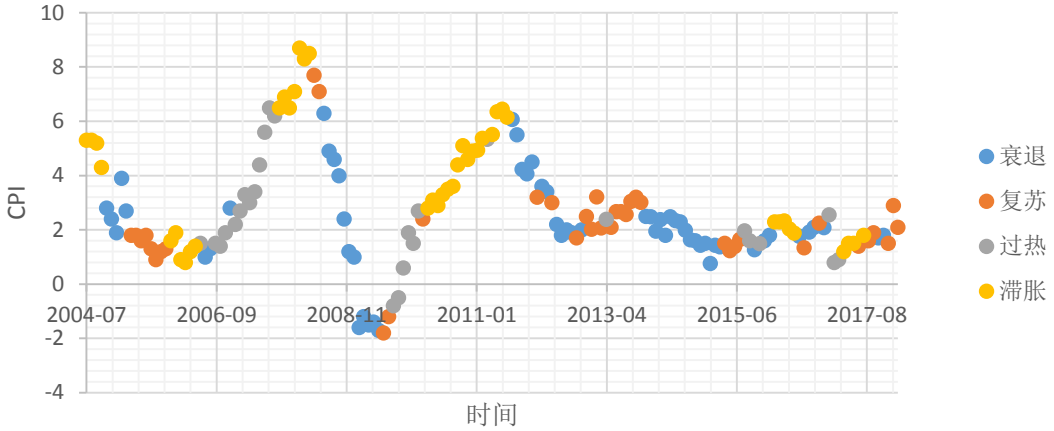

图 3.11　2004 年-2018 年经济周期轮动

四．模型构建

（一）XGBoost 参数设置

XGBoost 需要调整的参数包括：

1. n_estimator，　XGBoost 模型生成树的棵树。
2. eta(learning rate)，学习效率，通过减少每一步的权重，可以提高模型的鲁棒性
3. min_child_weight，决定最小叶子的权重和，用于避免过拟合，当它的值过大时，可以避免模型学习到局部的特殊样本。但是，当这个值过高，会导致欠拟合，可以通过交叉验证来调整。
4. max_depth，树的最大深度，也是用来避免过拟合的。max_depth 越大，模型

会学到更加局部的样本，可以通过交叉验证函数调优。

5. gamma，在节点分裂时，只有分裂后损失函数的值下降了，才会分裂这个节点。gamma 指定了节点分裂所需的最小损失函数下降值。这个参数的值越大，算法越保守。这个参数的值和损失函数息息相关，所以是需要调整的。
6. Subsample，这个参数控制对于每棵树，随机采样的比例。减小这个参数的值，算法会更加保守，避免过拟合。但是，如果这个值设置得过小，它可能会导致欠拟合。
7. Colsample_bytree，用来控制每棵随机采样的列数的占比。

（二）参数设置过程

1. 先设置学习效率为 0.1，通过数据实验选出最佳 n_estimator(决策树的数目)，采用总体预测准确率的平均值作为实验的评价。

首先固定其他参数不变，得到参数 n_estimator 与预测准确率的关系如下图所示。下图的纵坐标是 XGBoost 模型的预测准确率，横坐标为决策树的数目。从图中可以看出，准确率开始时随着决策树的数量增大而逐渐递增，当增大到 250 棵左右的时候逐步平稳，数值上表现为 AUC 的提升不足 0.1%。故选取参数 n_estimator 为 250。

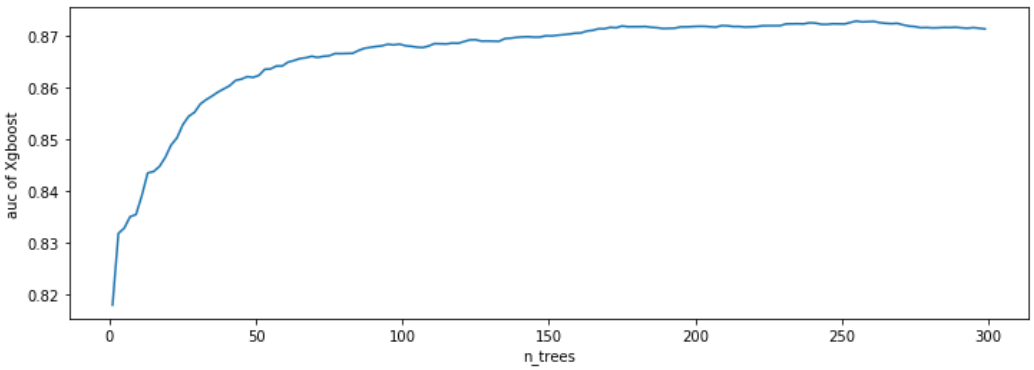

图 4.1 决策树数量取值与 AUC 之间的关系

2. Max_depth 和 min_child_weight 参数调优

确定决策树的数量等于 250 不变，采用网格贪心算法现在较大的范围里寻找最大化 AUC 的参数 max_depth 和 min_child_weight，确定大范围后，再在小范围内逐步调优。如下表所示，得到 max_depth 和 min_child_weight 的最优参数为 3，3。

表 4.1　max_depth 和 min_child_weight 调参过程

mean: 0.85439,	std: 0.01115,	params: {'max_depth': 3,	'min_child_weight': 1}
mean: 0.85532,	std: 0.00910,	params: {'max_depth': 3,	'min_child_weight': 3}
mean: 0.85526,	std: 0.01119,	params: {'max_depth': 3,	'min_child_weight': 5}
mean: 0.84564,	std: 0.01151,	params: {'max_depth': 5,	'min_child_weight': 1}
mean: 0.84730,	std: 0.01021,	params: {'max_depth': 5,	'min_child_weight': 3}
mean: 0.84658,	std: 0.00911,	params: {'max_depth': 5,	'min_child_weight': 5}
mean: 0.83964,	std: 0.00947,	params: {'max_depth': 7,	'min_child_weight': 1}
mean: 0.83926,	std: 0.01065,	params: {'max_depth': 7,	'min_child_weight': 3}
mean: 0.83894,	std: 0.00742,	params: {'max_depth': 7,	'min_child_weight': 5}
mean: 0.83838,	std: 0.01089,	params: {'max_depth': 9,	'min_child_weight': 1}
mean: 0.83526,	std: 0.01130,	params: {'max_depth': 9,	'min_child_weight': 3}
mean: 0.83540,	std: 0.01115,	params: {'max_depth': 9,	'min_child_weight': 5}

3.gamma，subsample，colsample_bytree 参数调优

该三个参数的调优方式与第二步的相同，最终得到的参数值分别为 0.1，8，8。

（三）XGBoost 建模

首先在利用 Bagging 的思想分别对离散化特征、归一化特征、热独编码特征和组合特征单独训练并预测时，为了减少运算量的同时能保证模型的精确度，采取对主要参数加入扰动的方式增加多样性，训练 100 次取平均值。

利用 XGBoost 模型画出决策树生成的过程。并根据每个特征所有节点上出现的频率或者每个特征对不纯度降低的贡献得到特征重要性序列。

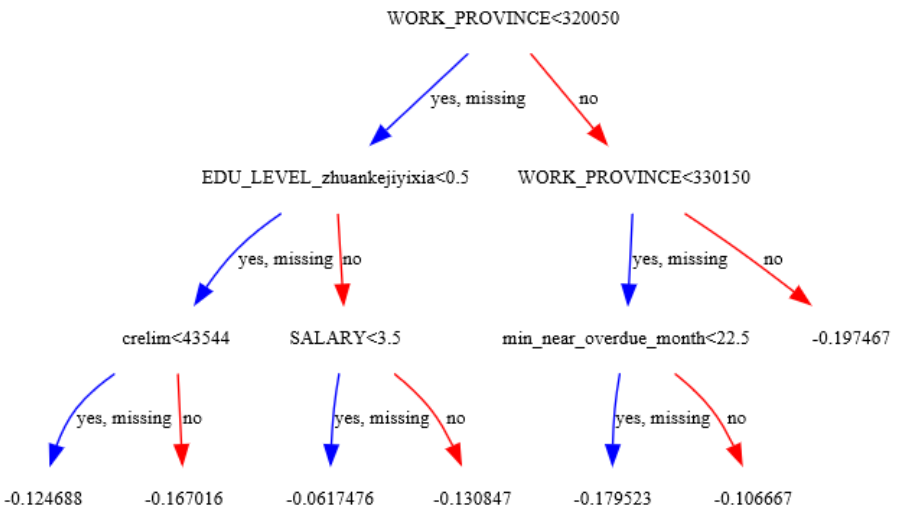

图 4.2　第一棵 XGBoost 决策树

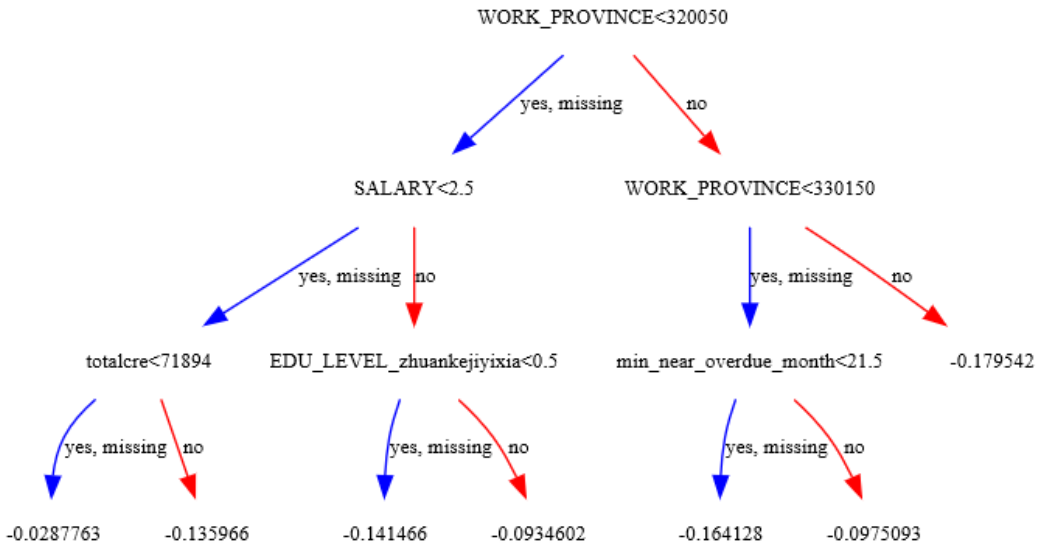

图 4.3 第二棵 XGBoost 决策树

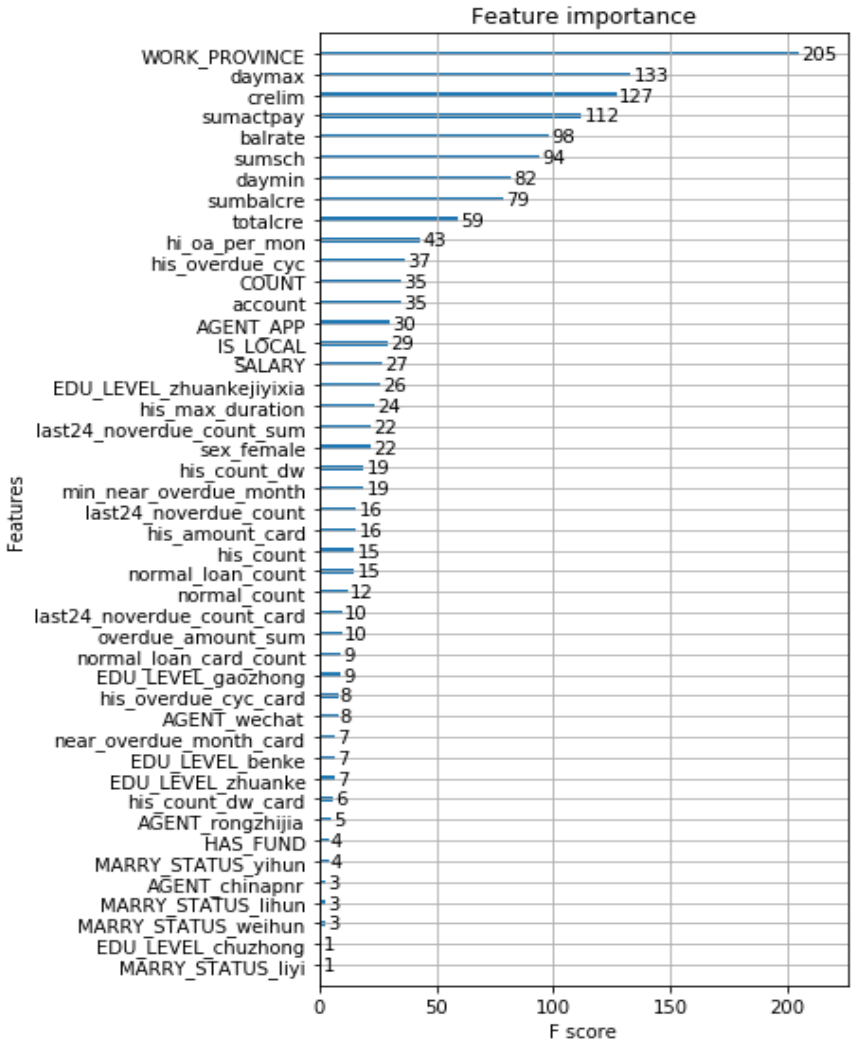

图 4.4 XGBoost 特征重要性序列

从 AUC 的衡量角度来看，所有模型的平均 AUC 都超过了 80%，说明集成树模型在信贷数据的预测上都具有良好的表现。其中，XGBoost 以 85.92%的 AUC 表现最佳

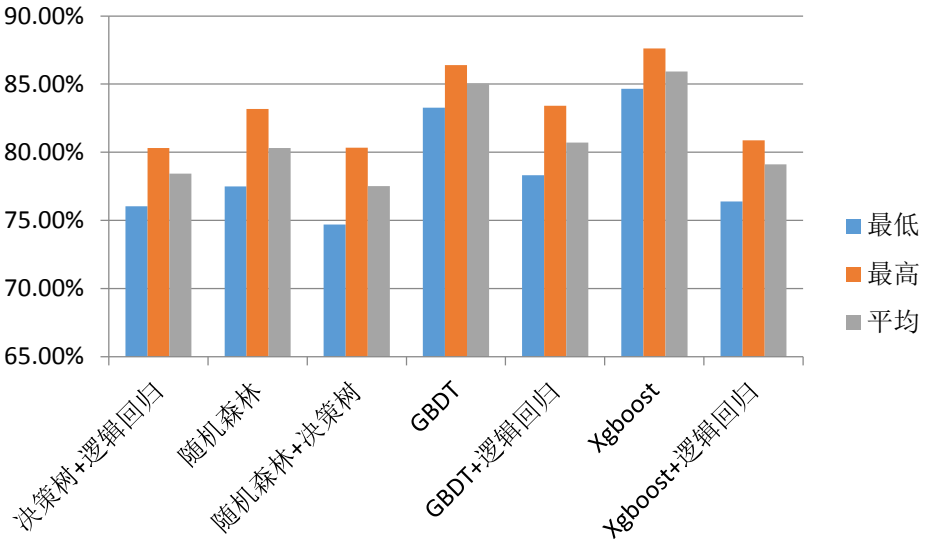

图 4.5 多种模型预测结果对比

(四)Local Interpretable Model-Agnostic Explanations

对于某客户甲模型得到的结论是接受贷款请求，原因是 WORK_PROVINCE，EDU_LEVEL_专科等因素支持该客户是一个“好”的客户。同理，对于客户乙模型得出的结论也是认为他是一个“好”的客户。

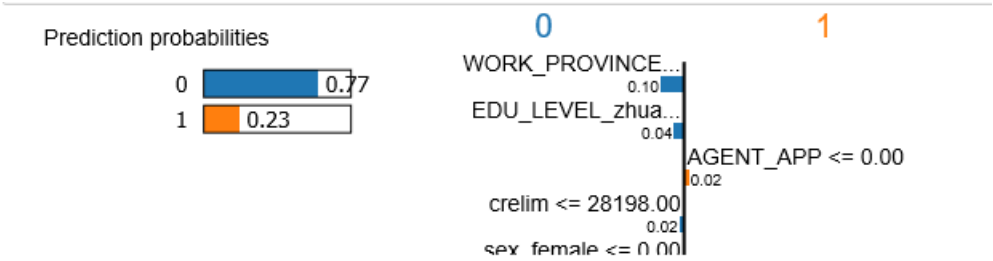

图 4.6 LIME 对客户甲做出决策的解释

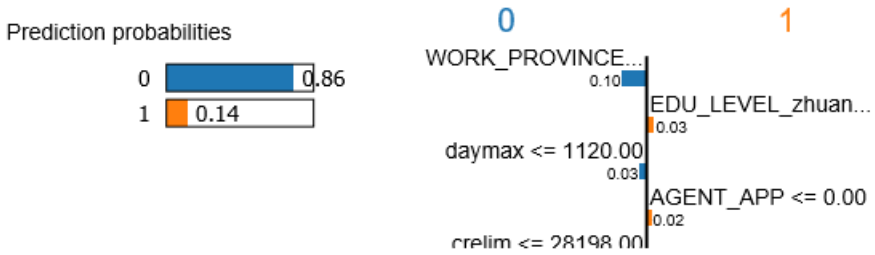

图 4.7 LIME 对客户乙做出决策的解释

(五)Stacking

本文先选用 XGBoost，随机森林和 GBDT 等模型作为基模型分别建立了信用预测模型，并使用 Logistic 回归模型作为第二层模型，将基模型的输出作为输入进行融合。采用 ROC 曲线和 AUC 作为模型表现的度量。实验的结果表明三个基模型的效果均明显由于单一的 Logistic 回归模型，其中 XGBoost 模型的效果最好。Stacking 融合模型的效果略差于 XGBoost，但是优于随机森林和 GBDT。

造成 Stacking 的效果比 XGBoost 模型差的原因还是在于基模型数量偏少，且都是以决策树为基础的集成树模型，基模型间的差异性较低，如果在基模型中加入支持向量机，神经网络，朴素贝叶斯等分类器模型，相信可以将 Stacking 模型的效果进一步提升。但是由于时间的限制，本文不再重复试验。

表 4.2　Stacking 与多种模型预测结果对比

模型	AUC			标准差
	最低	最高	平均	
决策树+逻辑回归	76.05%	80.32%	78.44%	0.008449
随机森林	77.48%	83.19%	80.31%	0.010341
随机森林+决策树	74.71%	80.34%	77.53%	0.010815
GBDT	83.26%	86.39%	85.04%	0.006012
GBDT+逻辑回归	78.32%	83.42%	80.72%	0.008439
XGBoost	84.65%	87.61%	85.92%	0.006511
XGBoost+逻辑回归	76.40%	80.88%	79.12%	0.00898
Stacking	84.51%	86.88%	85.63%	0.005969

(六)不平衡样本的 Ensemble-XGBoost 模型

由上文的模型之间的比较可以看出，在训练样本中 XGBoost 模型的表现要好于其他所有模型，故在测试集合上应用 XGBoost 模型。但是，由于训练集的 30000 条样本中，违约样本数仅占 6.25%，属于非平衡样本。如果采用误判率作为评估模型的效果，在本文的数据下可能出现较大的风险。一方面，预测的结果可能集中在较小的概率上，对于阈值的选择过于敏感；另一方面，若将所有的测试样本预测为不违约，则模型的准确率也将高达 93%以上，这种判断明显是没有意义的。常用的做法是使用 AUC 比较模型效果或者对拒真和受伪两种错误赋予不同的权重，通常“漏判一个坏人”与“冤枉一个好人”的代价比的取值为 5 或 10。

针对不平衡样本带来的风险，我们通常的做法有三种：其一是上采样，即复

制违约的样本使达到平衡。这种做法会严重影响模型的泛化能力，造成过拟合。比如在区分猫狗的分类问题中，大量复制猫的样本，若恰巧猫样本中黄猫居多，大规模复制的后果就是模型学习的方向变为区分黄色和其他。其二是下采样，即筛选出部分未违约的样本使达到平衡，但是这种方法会导致信息的丢失，其三使用 SMOTE 算法产生违约样本，但是该算法人为地产生过多的样本，引入了过多的主观因素。

本文提出了一种针对不平衡样本的 XGBoost 模型——Ensemble-XGBoost 模型。该模型先将训练集中违约样本提取出来，再在剩余的未违约样本中随机抽取与违约样本数量相同的样本合成新的训练子集，在子集上训练 XGBoost 模型。重复这样的操作 m 次后，我们得到由不同数据训练出的 m 个 XGBoost 模型，并对测试集的样本进行预测，将 m 个预测值加权平均（通常为等权）最为测试样本的预测违约概率。

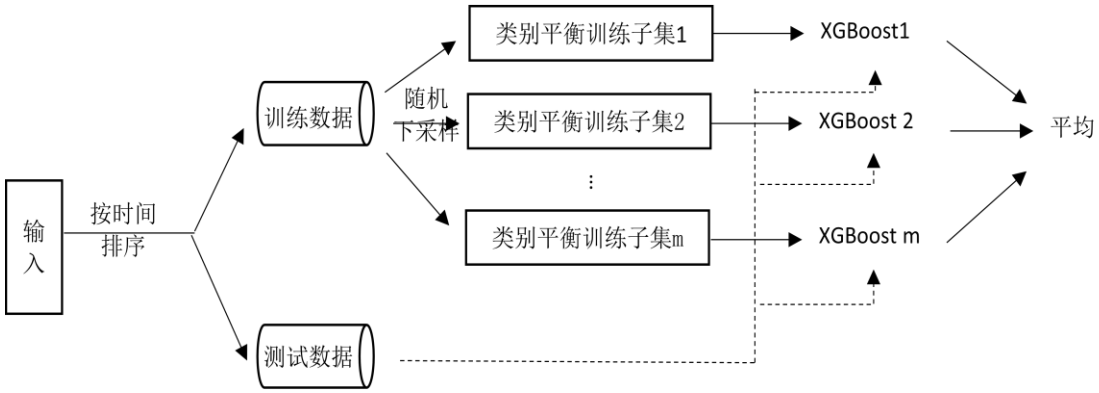

图 4.8　不平衡样本的 XGBoost 模型——Ensemble-XGBoost 模型

另外，模型在使用过程中需要注意，由于每个训练子集的样本是随机抽取的，也就意味着即便训练样本不变，训练两次得到的预测值可能不同。尽管大数定律和中心极限定理能够保证 m 足够大时预测值变化很小，但是对于某些介于违约、未违约之间的样本的预测结果可能有明显的区别。即可能出现一位先被预测为不会违约的客户可能在第二次预测中判定为违约这样的尴尬局面。因此为了避免这种情形，我们采取的做法是保留每次随机提取样本的随机数，即样本不变的条件下，每次 m 个训练子集得到的数据的相同的，即便训练数据有部分更新，用于训练子集的数据变化也不会明显，保证了模型结果稳定性。

进一步，不同 XGBoost 的差异性主要来源于训练数据的差异，但是为了丰富不同模型观察数据的角度，我们也可以对模型的特征和超参数进行采样，同样我们也需要保留采样的随机数。

类比于 FICO 的信用评分系统对于基于 Logistic Regression 模型预测出的违约概率转换为信用得分更便于理解，并将各个特征的得分展示出来，提高了模

型的可解释性。因此，本文在 Ensemble-XGBoost 模型得到的预测概率基础上，利用总体转换的方式，将概率转换为信用评分：

$$\text{score} = \ln\left(\frac{\text{p}}{1-\text{p}}\right) * \text{factor} + \text{offset} \quad (12)$$

其中，p 表示客户未违约的概率，factor 和 offset 为常数，用于将评分调整到合适的区间上，本文中采取 factor 为 20，offset 为 600。我们将测试样本的模型得分按顺序排列起来，选择合适的阈值，低于阈值的客户判断为可能违约，高于阈值的客户判断为不太可能违约。

表 4.3　测试样本的模型得分顺序排列

分数段	违约客户数	未违约客户数	总数	违约客户数累计占比	未违约客户数占比	坏账率
544-579	45	155	200	45.92%	8.15%	22.50%
580-592	23	177	200	69.39%	17.46%	11.50%
593-602	10	190	200	79.59%	27.44%	5.00%
603-614	8	192	200	87.76%	37.54%	4.00%
615-637	6	194	200	93.88%	47.74%	3.00%
638-652	3	197	200	96.94%	58.10%	1.50%
653-661	2	198	200	98.98%	68.51%	1.00%
662-669	0	200	200	98.98%	79.02%	0.00%
670-677	0	200	200	98.98%	89.54%	0.00%
678-710	1	199	200	100.00%	100.00%	0.50%

在实际的信用评分系统的使用中，经常利用 K-S 值来度量风控模型的性能：K-S 值通过“好”“坏”客户的累计百分比函数之间的最大距离度量模型的区分能力，距离越大则表明模型的区分能力越强。在本文的测试数据集上，计算得到的 K-S 值为 0.532，区分能力较高。

表 4.4　K-S 不同取值表示的模型区分能力

K-S 值	区分能力
<0.2	无
0.21-0.40	低
0.41-0.50	中
0.51-0.60	高
0.61-0.75	极高
>0.9	太高，可能有问题

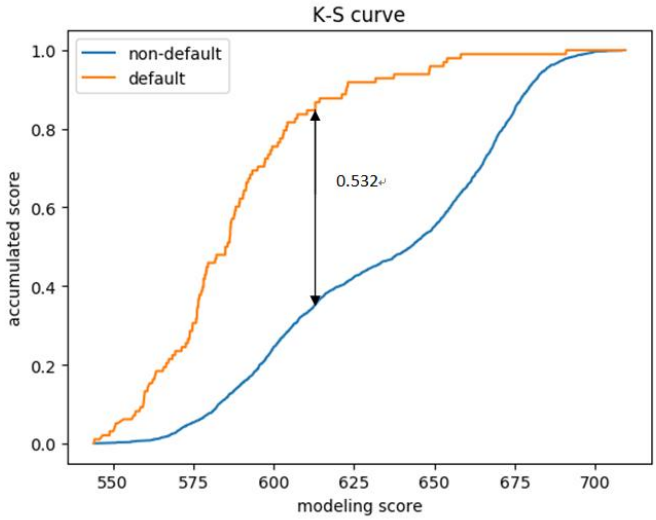

图 4.9　测试数据集得到的 K-S 值

(七)阈值选择

对于模型预测的结果的分类，即阈值的选择，我们通常采用的方法有两种：其一，最小化损失函数。先计算混淆矩阵，对于两种误判分别赋予不同的损失值，通常受伪的损失为拒真的 5-10 倍。其二，控制模型预测违约样本的个数，使得模型预测的违约样本数与实际违约样本数相近。本文综合考虑两种方法，对针对不平衡样本的 Ensemble-XGBoost 模型，将两种误判情况的损失比设为 5，通过最小化损失函数计算得到阈值为 0.8。

五．模型总结及创新点

(一) 模型总结

本文将集成树模型和融合模型应用在了信贷风险评估的问题中，并以东证期货公司给出的数据为基础，构造新的特征并进行了实证分析。实验结果发现，XGBoost 模型和 Stacking 模型要比其他模型具有更高的准确率和稳定性。其中，XGBoost 模型的 AUC 可以达到 86%，Stacking 模型的效果略差于 XGBoost 模型。但是 Stacking 可以有效利用基模型从原始数据中提取出的有效组合特征，能够预先分析出有效的特征、特征组合，充分利用历史数据避免忽略原始数据中的重要信息，从而提高模型精度。如果在 Stacking 的基模型中加入更多的分类器模型，可以进一步提升模型效果，甚至超过 XGBoost 模型。

然后，本文还对黑箱模型进行更加深入的解释和剖析。比如通过 XGBoost 模型绘制迭代过程每一颗决策树，并得到的特征重要性进行筛选和剔除特征。还利用局部可解释工具（LIME）对 XGBoost 模型针对预测样本作出的每一决策进行合理化解释 。

进一步，在得到 XGBoost 模型效果好于其他模型的基础上，我们针对数据不平衡样本的特性构造 Ensemble-XGBoost 模型，利用 Bagging 的思想，避免预测概率集中在某一区间上，降低了阈值的敏感性。并将此模型应用在检验数据集上，将预测概率转换为信用得分，利用损失函数和经济周期宏观变量调节阈值。结果表明 K-S 值为 0.532，可以应用在实际场合中。

最后，本文得出结论，XGBoost 模型的精度和稳定性要高于其他模型，而通过模型堆叠（Stacking）的方式还能进一步提升模型的效果。银行信贷数据具有指标多、信噪比低，不平衡的特点，使用 Ensemble-XGBoost 模型和模型堆叠技术构建个人信贷风险评估模型对实际应用具有重要指导意义。

注：使用本文建立的模型对测试集的预测结果见附件三。

（二）创新点

本文的创新点在于不仅仅应用信贷部门的数据构建模型，还利用客户在其他多家征信系统的信用记录通过特征过程的方式构造新的特征，并使用 WOE，IV 等评价方式筛选出有效的变量。同时，根据不同时间违约率变化的特征，结合美林时钟理论，我们将经济周期划分为衰退、复苏、过热和滞涨四个阶段，利用历史统计规律，对阈值进行不同程度上的调节。

另外，在模型的选择上，本文使用了近年来在工业界比较热门的集成模型——XGBoost 模型，XGBoost 模型属于集成学 Boosting 中的一种，通过不断对回归树的残差建模降低模型预测的偏差，在寻找最优解时利用到了损失函数的二阶导数，使得模型在效果上更加精确和稳定。此外 XGBoost 也因其能够并行运算而深得业界青睐。本文将 XGBoost 模型与诸多基于树的模型和与 Logistic 回归复合的模型进行对比，得到其效果确实优于其他模型。

为了更进一步的提升模型效果，本文还考虑使用堆叠技术（Stacking）融合包含 XGBoost，随机森林，GBDT 在内的多种分类器模型的输出结果，训练第二层的 Logistic 模型。尽管最终 Stacking 的效果略逊色于 XGBoost 模型，但是如果提高第一层基础分类器的数量，比如加入支持向量机、神经网络、朴素贝叶斯模型等，是有可能突破 XGBoost 模型的效果。最后，基于不平衡样本的特征，我们引入 Ensemble-XGBoost 模型，利用 Bagging 的思想，避免预测概率集中在某一区间上，降低了阈值的敏感性。此外，为了弥补“黑箱”模型造成的解释性降低

的问题，本文尽可能的描述 XGBoost 模型的建树过程，以及使用 LIME（局部模型解释工具)对模型做出的决策进行解释，使得模型的实用性和可靠性大大提升。

六．参考文献

[1] He X. Pan J, Jin O. et al. Practical Lessons from Predicting Clicks on Ads at Facebook [M]. ACM,2014.

[2] Rohabi R. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants[J]. Machine Learning, 1999, 36: 105-139,

[3] Chen T Q, Li H, Yang Q, et al. General Function Matrix Factorization Using Gradient Boosting[J]. Proceedings of the 30th International Conference on Machine Learning, 2013, 28: 436-444.

[4] Marco T., Sameer S. "Why Should I Trust You?" Explaining the Predictions of Any Classifier[J]. arXiv:1602.04938v3.

[5] J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29:1189-1232, 1999.

[6] Morris N L, Price P O. The Dodd-Frank Wall Street Reform and Consumer Protection Act[J]. Cpa Journal, 2010, 3(3):77–78.

[7] Magee J R. Peer-to-Peer Lending in the United States: Surviving after Dodd-Frank[J]. N.c.banking Inst, 2011.

[8] Laitinen E K. Predicting a corporate credit analyst's risk estimate by logistic and linear models[J]. International Review of Financial Analysis, 1999, 8(99):97–121.

[9] Malekipirbazari M, Aksakalli V. Risk assessment in social lending via random forests[J]. Expert Systems with Applications, 2015, 42(10):4621-4631.

[10] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016:785-794.

[11] 蔡文学，罗永豪，张冠湘，钟慧玲. 基于 GBDT 与 Logistic 回归融合的个人信贷风险评估模型及实证分析[J]. 财政金融，2017, 02（2）：1-4.

[12] 张奇，胡蓝艺，王钰. 基于 Logit 与 SVM 的银行业信用风险预警模型研究[J]. 系统工程理论与实践，2015（07）：14-18.

[13] 白鹏飞，安琪. 基于多模型融合的互联网信贷个人信用评估方法[J]. 华南师范大学学报，2017，49（6）：119-123

[14] 刘新海. 运用大数据开展P2P信用风险评估的Upstart[J]. 征信,2016(6)：18-22.

[15] 冯科, 何理. 互联网消费金融的创新[J]. 中国金融, 2016(11):32-34.

[16] 石庆焱. 个人信用评分模型及其应用[M]. 中国方正出版社, 2006.

[17] 王春峰, 万海晖. 商业银行信用风险评估及其实证研究[J]. 管理科学学报, 1998(1):68-72.

[18] 于立勇, 詹捷辉. 基于 Logistic 回归分析的违约概率预测研究[J]. 财经研究, 2004, 30(9):15-23.

[19] 汪办兴. 我国商业银行信用风险模型的国际比较与改进[J]. 当代经济科学, 2007, 29(3):65-70.
[20] 陈启伟，王伟. 基于 Ext-GBDT 集成的类别不平衡信用评分模型[J]. 计算机应用研究，2018，35（2）：421-427.
[21] 孙云，张昊旻. 美林投资时钟理论在中国金融市场应用探索[J]. 经济问题探索，2015（9）：57-64.

七. 附录

(一)统计软件

本文的数据处理以及模型构造主要使用了 Python, SAS 两种统计软件。其中数据处理主要使用 SAS，模型构造使用了 Python。

(二)最终变量汇总表

逾期	curr_overdue_cyc_max	信贷账户当前最大逾期期数（贷款）
	curr_overdue_cyc_card_max	信贷账户当前最大逾期期数（贷记卡）
	curr_overdue_amount_card_sum	信贷账户当前逾期总金额（贷记卡）
	overdue_amount_sum	信贷账户当前逾期总金额
	normal_loan_count	信贷账户当前逾期总金额
	normal_loan_card_count	当前状态正常的信贷账户数量（贷款）
	normal_count	当前状态正常的信贷账户总数量
	his_count_dw	征信记录上信贷账户的逾期总笔数（贷款）
	his_overdue_cyc	征信记录上信贷账户的最大逾期期数（贷款）
	his_max_duration	征信记录上信贷账户的单月最高逾期总额（贷款）
	hi_oa_per_mon	征信记录上信贷账户的最大贷款时长（贷款）
	his_count_dw_card	征信记录上信贷账户的逾期总笔数（贷记卡）
	his_overdue_cyc_card	征信记录上信贷账户的最大逾期期数（贷记卡）
	his_amount_card	征信记录上信贷账户的逾期总金额（贷记卡）
	his_count	征信记录上信贷账户的逾期总笔数
	last24_overdue_count	征信记录上过去 24 个月有过 1 个月及以上逾期的账户数（贷款）

	last24_noverdue_count	征信记录上过去 24 个月从未有过逾期的账户数（贷款）
	last24_od_cyc_max	征信记录上过去 24 个月最大逾期期数（贷款）
	near_overdue_month_card	征信记录上最近一次 1 个月及以上逾期距查询日期的月份数（贷记卡）
	last24_noverdue_count_card	征信记录上过去 24 个月从未有过逾期的账户数（贷记卡）
	min_near_overdue_month	征信记录上最近一次 1 个月及以上逾期距查询日期的月份数
	last24_noverdue_count_sum	征信记录上过去 24 个月从未有过逾期的账户数
个人基本信息	Y	目标变量值
	ID_CARD	身份证
	LOAN_DATE	放款时间
	AGENT	客户渠道
	IS_LOCAL	是否本地籍
	WORK_PROVINCE	工作城市
	EDU_LEVEL	教育水平
	MARRY_STATUS	婚姻状态
	SALARY	收入
	HAS_FUND	是否有公积金
账户账龄	daymax	信贷账户最大账龄（单位天数）
	daymin	信贷账户最小账龄（单位天数）
	COUNT	信贷账户个数
	sumbalcre	信贷账户本金余额
授信余额	totalcre	信贷账户授信总额
	sumsch	信贷账户当前应还金额
	sumactpay	信贷账户实际还款金额
	balrate	信贷账户额度使用率
	account	贷款账户个数
	crelim	贷款账户授信总额

(三)模型主要代码

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jan 29 14:54:14 2018

@author: ling
"""
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import (RandomTreesEmbedding, RandomForestClassifier,
                                        GradientBoostingClassifier)
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve,auc
from sklearn.pipeline import make_pipeline
from sklearn import metrics
from sklearn.datasets    import    make_hastie_10_2
from xgboost.sklearn import XGBClassifier
import xgboost as xgb
import os
path = r'C:\Users\ling\Desktop\datanew.csv'
data = pd.read_csv(path, header = 0,encoding = 'cp936')
data.index = data['REPORT_ID']
del(data['REPORT_ID'])

#提取性别变量
idcard = list(data['ID_CARD'])
SEX    =list()
for i in range(len(idcard)):
      SEX.append(idcard[i][16:17])
sex = list()
for i in range(len(idcard)):
      if int(SEX[i])%2 ==1:
            sex.append('male')
      else:
            sex.append('female')
data['sex'] = sex
      Y = data['Y']
data['IS_LOCAL']=data['IS_LOCAL'].replace({'本地籍':0,'非本地籍':1})
data = data.fillna(0)
X = data
del(X['Y'])
del(X['ID_CARD'])
del(X['LOAN_DATE'])
X = pd.get_dummies(X)
del(X['sex_male'])
n_estimator = 125
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3)
```

```
X_train, X_train_lr, y_train, y_train_lr = train_test_split(X_train,y_train,test_size=0.3)
rt = RandomTreesEmbedding(max_depth=3, n_estimators=n_estimator,
    random_state=0)
rt_lm = LogisticRegression()
pipeline = make_pipeline(rt, rt_lm)
pipeline.fit(X_train, y_train)
y_pred_rt = pipeline.predict_proba(X_test)[:, 1]
fpr_rt_lm, tpr_rt_lm, _ = roc_curve(y_test, y_pred_rt)

# Supervised transformation based on random forests
rf = RandomForestClassifier(max_depth=3, n_estimators=n_estimator)
rf_enc = OneHotEncoder()
rf_lm = LogisticRegression()
rf.fit(X_train, y_train)
rf_enc.fit(rf.apply(X_train))
rf_lm.fit(rf_enc.transform(rf.apply(X_train_lr)), y_train_lr)
y_pred_rf = rf.predict_proba(X_test)[:, 1]
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_pred_rf)
y_pred_rf_lm = rf_lm.predict_proba(rf_enc.transform(rf.apply(X_test)))[:, 1]
fpr_rf_lm, tpr_rf_lm, _ = roc_curve(y_test, y_pred_rf_lm)
grd = GradientBoostingClassifier(n_estimators=n_estimator)
grd_enc = OneHotEncoder()
grd_lm = LogisticRegression()
grd.fit(X_train, y_train)
grd_enc.fit(grd.apply(X_train)[:, :, 0])
grd_lm.fit(grd_enc.transform(grd.apply(X_train_lr)[:, :, 0]), y_train_lr)
y_pred_grd_lm = grd_lm.predict_proba(
    grd_enc.transform(grd.apply(X_test)[:, :, 0]))[:, 1]
fpr_grd_lm, tpr_grd_lm, _ = roc_curve(y_test, y_pred_grd_lm)

# The gradient boosted model by itself
y_pred_grd = grd.predict_proba(X_test)[:, 1]
fpr_grd, tpr_grd, _ = roc_curve(y_test, y_pred_grd)

#Xgboost
clf = XGBClassifier(
    n_estimators=125,#三十棵树
    learning_rate =0.1,
    max_depth=3,
    min_child_weight=3,
    gamma=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    objective= 'binary:logistic',
```

```
        nthread=4,
        scale_pos_weight=1,
        reg_lambda=1,
        seed=27)

#model_bst = xgb.train(params, d_train, 30, watchlist, early_stopping_rounds=500,
verbose_eval=10)
model_sklearn=clf.fit(X_train, y_train)

#y_bst= model_bst.predict(d_test)
y_pred_xgb= clf.predict_proba(X_test)[:,1]
fpr_xgb, tpr_xgb, _ = roc_curve(y_test, y_pred_xgb)

'''
auc_xgb = list()
for i in np.arange(1,200):
        n_estimator = i
        clf = XGBClassifier(
                n_estimators=n_estimator,
                learning_rate =0.1,
                max_depth=3,
                min_child_weight=1,
                gamma=0.3,
                subsample=0.8,
                colsample_bytree=0.8,
                objective= 'binary:logistic',
                nthread=4,
                scale_pos_weight=1,
                reg_lambda=1,
                seed=27)
        model_sklearn=clf.fit(X_train, y_train)
        y_pred_xgb= clf.predict_proba(X_test)[:,1]
        fpr_xgb, tpr_xgb, _ = roc_curve(y_test, y_pred_xgb)
        auc_xgb.append(auc(fpr_xgb, tpr_xgb))

plt.plot(np.arange(1,200),auc_xgb)
plt.show()
'''

#Xgboost+lm
xgb = clf
xgb_enc = OneHotEncoder()
xgb_lm = LogisticRegression()
xgb.fit(X_train, y_train)
```

```
xgb_enc.fit(xgb.apply(X_train))
xgb_lm.fit(xgb_enc.transform(xgb.apply(X_train_lr)), y_train_lr)
y_pred_xgb_lm = xgb_lm.predict_proba(
    xgb_enc.transform(xgb.apply(X_test)))[:, 1]
fpr_xgb_lm, tpr_xgb_lm, _ = roc_curve(y_test, y_pred_xgb_lm)

#plt.figure(4)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_rt_lm, tpr_rt_lm, label='RT + LR(aera = {0:0.4f})'''.format(auc(fpr_rt_lm, tpr_rt_lm)))
plt.plot(fpr_rf, tpr_rf, label='RF(aera = {0:0.4f})'''.format(auc(fpr_rf, tpr_rf)))
plt.plot(fpr_rf_lm, tpr_rf_lm, label='RF + LR(aera = {0:0.4f})'''.format(auc(fpr_rf_lm, tpr_rf_lm)))
plt.plot(fpr_grd, tpr_grd, label='GBT(aera = {0:0.4f})'''.format(auc(fpr_grd, tpr_grd)))
plt.plot(fpr_grd_lm, tpr_grd_lm, label='GBT + LR(aera = {0:0.4f})'''.format(auc(fpr_grd_lm,
tpr_grd_lm)))
plt.plot(fpr_xgb, tpr_xgb, label='Xgb(aera = {0:0.4f})'''.format(auc(fpr_xgb, tpr_xgb)))
plt.plot(fpr_xgb_lm, tpr_xgb_lm, label='Xgb+LR(aera = {0:0.4f})'''.format(auc(fpr_xgb_lm,
tpr_xgb_lm)))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()

#Xgboost features importance
from xgboost import plot_tree
from xgboost import plot_importance
import matplotlib.pyplot as plt
from graphviz import Digraph
import pydot
os.environ["PATH"] += os.pathsep + 'D:/graphviz/bin/'
def my_plot_importance(booster, figsize):
    from matplotlib import pyplot as plt
    from xgboost import plot_importance
    fig, ax = plt.subplots(1,1,figsize=figsize)
    return plot_importance(booster=booster, ax=ax)
my_plot_importance(model_sklearn, (5,10))
plt.show()
auc_rt_LR = list()
auc_rf = list()
auc_rf_LR = list()
auc_GBDT = list()
auc_GBDT_LR = list()
auc_xgboost = list()
auc_xgboost_LR = list()
```

```
for i in range(100):
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3)
    X_train, X_train_lr, y_train, y_train_lr = train_test_split(X_train,y_train,test_size=0.3)
    rt = RandomTreesEmbedding(max_depth=3, n_estimators=n_estimator,
    random_state=0)

    rt_lm = LogisticRegression()
    pipeline = make_pipeline(rt, rt_lm)
    pipeline.fit(X_train, y_train)
    y_pred_rt = pipeline.predict_proba(X_test)[:, 1]
    fpr_rt_lm, tpr_rt_lm, _ = roc_curve(y_test, y_pred_rt)

    # Supervised transformation based on random forests
    rf = RandomForestClassifier(max_depth=3, n_estimators=n_estimator)
    rf_enc = OneHotEncoder()
    rf_lm = LogisticRegression()
    rf.fit(X_train, y_train)
    rf_enc.fit(rf.apply(X_train))
    rf_lm.fit(rf_enc.transform(rf.apply(X_train_lr)), y_train_lr)
    y_pred_rf = rf.predict_proba(X_test)[:, 1]
    fpr_rf, tpr_rf, _ = roc_curve(y_test, y_pred_rf)
    y_pred_rf_lm = rf_lm.predict_proba(rf_enc.transform(rf.apply(X_test)))[:, 1]
    fpr_rf_lm, tpr_rf_lm, _ = roc_curve(y_test, y_pred_rf_lm)
    grd = GradientBoostingClassifier(n_estimators=n_estimator)
    grd_enc = OneHotEncoder()
    grd_lm = LogisticRegression()
    grd.fit(X_train, y_train)
    grd_enc.fit(grd.apply(X_train)[:, :, 0])
    grd_lm.fit(grd_enc.transform(grd.apply(X_train_lr)[:, :, 0]), y_train_lr)
    y_pred_grd_lm = grd_lm.predict_proba(
        grd_enc.transform(grd.apply(X_test)[:, :, 0]))[:, 1]
    fpr_grd_lm, tpr_grd_lm, _ = roc_curve(y_test, y_pred_grd_lm)

    # The gradient boosted model by itself
    y_pred_grd = grd.predict_proba(X_test)[:, 1]
    fpr_grd, tpr_grd, _ = roc_curve(y_test, y_pred_grd)

    #Xgboost
    clf = XGBClassifier(
        n_estimators=250,#三十棵树
        learning_rate =0.1,
        max_depth=3,
        min_child_weight=5,
        gamma=0.2,
```

```
        subsample=0.6,
        colsample_bytree=0.9,
        objective= 'binary:logistic',
        nthread=4,
        scale_pos_weight=1,
        reg_lambda=1,
        seed=27)

    #model_bst = xgb.train(params, d_train, 30, watchlist, early_stopping_rounds=500,
verbose_eval=10)
    model_sklearn=clf.fit(X_train, y_train)

    #y_bst= model_bst.predict(d_test)
    y_pred_xgb= clf.predict_proba(X_test)[:,1]
    fpr_xgb, tpr_xgb, _ = roc_curve(y_test, y_pred_xgb)
    xgb = clf
    xgb_enc = OneHotEncoder()
    xgb_lm = LogisticRegression()
    xgb.fit(X_train, y_train)
    xgb_enc.fit(xgb.apply(X_train))
    xgb_lm.fit(xgb_enc.transform(xgb.apply(X_train_lr)), y_train_lr)
    y_pred_xgb_lm = xgb_lm.predict_proba(
        xgb_enc.transform(xgb.apply(X_test)))[:, 1]
    fpr_xgb_lm, tpr_xgb_lm, _ = roc_curve(y_test, y_pred_xgb_lm)
    auc_rt_LR.append(auc(fpr_rt_lm, tpr_rt_lm))
    auc_rf.append(auc(fpr_rf, tpr_rf))
    auc_rf_LR.append(auc(fpr_rf_lm, tpr_rf_lm))
    auc_GBDT.append(auc(fpr_grd, tpr_grd))
    auc_GBDT_LR.append(auc(fpr_grd_lm, tpr_grd_lm))
    auc_xgboost.append(auc(fpr_xgb, tpr_xgb))
    auc_xgboost_LR.append(auc(fpr_xgb_lm, tpr_xgb_lm))
```