西安交通大学

毕业设计（论文）

题　目　深度 Boosting 决策树算法在欺诈检测中的应用

管理　学院　工业工程　专业　工业工程 71　班

学生姓名　周梦豪

学　　号　2174111346

指导教师　王尧

设计所在单位　西安交通大学大学管理学院

2021　年　06　月

摘 要

在我国数字经济繁荣的过程中，出现了欺诈数量的激增和新的欺诈手段的不断衍生，给企业和用户带来了巨大的经济损失。欺诈呈现出的新特征使得传统的欺诈检测方式日益暴露出不足。因此，如何开发更高效的欺诈检测方法成为企业管理者和研究人员共同关心的问题。随着以深度学习为代表的机器学习方法在各个领域显示出巨大的应用潜力，采用机器学习方法进行欺诈检测成为新的研究思路。

本文探究了一种新的机器学习方法-深度Boosting 决策树算法在欺诈检测中的应用。本文首先构建了一种融合神经网络与决策树的模型软决策树，并将其作为弱分类器与 Boosting 框架结合形成了深度 Boosting 决策树算法，接着选取了4种Boosting 算法（AdaBoost 、XGBoost 、SMOTEBoost 和RUSBoost ）作为基准模型，在3个真实欺诈数据集上与深度Boosting 决策树进行了性能比较；然后本文探究了超参数对模型性能的影响；最后尝试打开模型，对模型进行解释，使用OBB 方法在一个互联网金融借贷数据集上得到了特征重要性排序。

实验结果显示深度Boosting 决策树算法整体性能优于这 4 种Boosting 算法；软决策树的数量T和决策树的深度d均会提升模型性能，且二者存在交互作用；神经网络的层数c对模型性能的提升有限，相反过多的层数会带来容易过拟合，性能不稳定的问题；根据特征重要性排序发现了一种在过去被人们忽略的一种重要特征 inq.last.6mths，同时还发现目前的借贷标准无法有效地区分客户群体，降低欺诈风险。

关键字：欺诈检测；Boosting 算法；超参数；特征重要性排序

ABSTRACT

The past decade has seen increasingly rapid advances in the field of digital economy, which also leads to the surge of fraud cases and the application of new fraud tools. It is imperative to reduce the damage caused by fraud. However, the performance of traditional fraud detection methods is limited by the new features of fraud. Therefore, developing more efficient fraud detection model has become an issue cared by both enterprise managers and researchers. Fortunately, the success of machine learning methods in many fields inspires us to apply them to fraud prediction.

In this paper, we demonstrate the value of applying a state-of-the-art machine learning Method called Deep boosting Decision Tree to fraud prediction. To assess the performance of our model, we select four boosting modeling including AdaBoost, XGBoost, SMOTEBoost and RUSBoost as benchmarks and compare their performance on three real fraud datasets. We also explore the how the hyper-parameters affect the model performance. Finally, we seek to open up the model and obtain the feature importance through OOB method.

The results show that our Deep Boosting Decision Tree Model outperforms the four benchmarks. Both the number of soft decision trees and the depth of decision trees can improve the model performance, and there is an interaction between them. The number of layers of the neural network does not do anything for the improvement of model performance. On the contrary, too many layers will result in over-fitting and unstable performance. According to the ranking of feature importance, we find an important feature-inq.last.6mths, which was ignored by people in the past, and also find an overlooked customer group

KEY Words: fraud detection; Boosting model; hyper-parameter; feature importance

目 录

1　绪论

1.1 欺诈检测简述

移动互联网的普及极大地改变了传统行业，尤其是与金融、购物行业的不断深度融合，更是催生了数字金融和电子商务的商业模式。这个过程不仅给使用者带来了更好的使用体验，而且极大地推动了我国数字经济的繁荣。截至 2020 年我国数字经济规模已达到 39.2 万亿元，占 GDP 比重为 38.6%[1]。但在这个过程中也出现了一些问题，例如新的欺诈手段不段出现，尤其是跟互联网金融相关的欺诈呈现出专业化、产业化、隐蔽化、场景化的特征[2]。欺诈不仅会给用户和公司带来严重的经济损失，而且会严重阻碍我国数字经济的健康规范发展。

由于数字技术的快速发展，导致欺诈手段的日益专业化，隐蔽化，传统的欺诈检测手段如专家策略，名单库等，在面对这些新的欺诈手段时往往出现效率低，精度低，维度单一等弊端。因此，探索新的更高效的欺诈检测方式成为工业界和学术界共同关注的话题。

近年来，机器学习方法在很多应用领域均取得了很大的成功，并展现出巨大的应用潜力潜力，因此，采用机器学习方法进行欺诈检测成为新的研究思路，并初步取得了一些成绩，例如 Bao Y 等人探究了 RusBoost 算法在上市公司财务欺诈检测中的应用[3]。但是在采用传统的机器学习方法时也会出现一些问题，如在面对具体的欺诈检测场景时，往往出现效率低，可解释性差等问题。

因此，针对欺诈检测问题，开发更高效，可解释性更好的机器学习算法对我国数字经济的健康发展具有十分重要的意义。

1.2 本文研究问题及工作内容

1.2.1 研究问题

本文的研究的问题是如何更高效地进行欺诈检测，由于从机器学习的视角可以将欺诈检测问题看作非均衡分类问题，因此本文的研究思路是从机器学习中的非均衡分类出发，探究一种新的机器学习方法在处理非均衡分类问题的效果，并将其应用在欺诈检测问题上。因此，本文研究的问题包括：

1) 如何设计新的算法解决非均衡分类问题
2) 如何评价算法的性能
3) 如何对训练出来的模型进行解释

1.2.2 工作内容

本文的研究目的是探究深度Boosting 决策树算法(将软决策树与Boosting 框架结合)在欺诈检测中的应用。因此本文的主要工作包括以下几个方面：

1) 文献综述，主要是总结近些年学者们对非均衡分类分类问题的解决方法。

2) 算法设计，包括：完成软决策树的构建，解决软决策树与Boosting 框架融合的关键问题，完成优化方法的设计，总结模型学习过程。

3) 算法实现与比较，思路是选取多种Boosting 算法并和合适的评价指标，比较不同算法在若干真实的欺诈交易数据集上的性能。

4) 超参数调整，探究模型众多超参数对模型性能的影响，总结超参数确定的规律。

5) 模型解释，主要是结合数据集本身的业务背景，尝试对模型结果做出解释，并能够对企业管理者产生启发。

1.3 本文框架

在第 2 章中，本文从处理非均衡问题的三个角度、改进的 Boosting 算法和软决策树等方面进行了文献综述。

在第 3 章中，本文构建了软决策树模型，并将其与 Boosting 框架进行了结合，解决了数据重构与结合策略，此外又从前向分布算法的视角推导了分步优化策略。

在第 4 章中，本文在三个真实数据集上训练了五种模型，并对其结果进行了比较，并进行了假设检验分析了其性能差异的显著性。

在第 5 章中，本文探究了众多模型超参数对模型性能的影响，包括：软决策树的数量、深度和神经网络的层数。

在第 6 章中，本文尝试打开训练得到的模型，获得了特征重要性排序，并结合管理实际进行了分析。

在第 7 章中，本文对文章的内容进行了总结包括：本文的结论与局限，以及对未来研究方向的一个展望。

2 文献综述

2.1 非均衡分类问题

虽然许多机器学习方法可以运用到欺诈检测中，如分类、聚类、关联分析以及异常诊断[4]，但从机器学习的视角来看，欺诈检测本质上是非均衡分类问题，因此本文研究将从非均衡分类问题出发。非均衡分类问题是指不同类别的样本数量相差悬殊的一种分类问题[5]。在欺诈检测问题中，大多数的交易是正常的，属于多数类，而欺诈交易是相对较少的，属于少数类。欺诈检测的目的就是将交易进行分类，识别出异常少数类交易，因此我们可以从对非均衡分类问题的研究中获取解决欺诈检测问题的思路。

非均衡性主要体现在两个方面：1、类间不均衡，即某类样本数量远远少于其他类，这是导致分类性能不佳的主要原因；2、类内不均衡，即某一类中间会存在次子集，有研究表明类内不均衡也会影响分类性能[6]。本文研究的是欺诈检测问题，非均衡性主要体现在类间不均衡上，因此，本文假设要研究的欺诈检测问题仅存在类间不均衡性，不存在类内不均衡性。

传统机器学习分类方法无法解决类间不均衡性问题的原因是，传统的分类算法在被研究者们提出时，往往会假设对不同类别错误分类的代价相同，因此在应用这些分类算法进行欺诈检测时，会造成模型过于“关注”多数类的信息，忽略少数类的信息，因此会出现多数类信息掩盖少数类信息，无法准确识别少数类的问题，从而造成传统模型在少数类上往往出现欠拟合现象，错误率较高。而在现实场景中人们有时会更关注模型在少数类上的结果，例如在欺诈检测中，欺诈属于少数类，企业管理者往往更关心在少数类上模型的精度，因此要开发高效的欺诈检测模型，必须要解决模型在少数类上性能不佳的问题。

为了解决上述问题，许多学者从不同角度给出了解决方案。这些方法大致可以从数据、算法和评价指标三个层次进行划分。

2.1.1 数据层面

由于原始训练数据是非均衡的，要解决该问题的第一个思路就是在数据层面采用某种策略获得一个相对均衡的数据集，在此基础上再用来训练模型。采用的策略通常包括两类：

1) 过采样策略

该策略的思想是根据少数类样本生成一些新的少数类数据，从而增加少数类样本数量。常用的过采样方法包括：1、随机过采样，即随机选择一些少数类样本进行复制，将其加入少数类中，有研究表明这种方法并不能有效解决过拟合的问题[7]；2、基于人工合成新数据的过采样方法(SMOTE)，即使用KNN求得少数类的“近邻”数据，在这些近邻数据之间进行随机插值[8]，获得新数据。

2) 欠采样策略

该策略的思想是根据多数类样本丢弃一些数据或者合成一些更少的数据来代替原来数据，从而减少多数类样本数量。常用的欠采样方法包括：1、随机欠采样，即随机选择一些多数类样本进行丢弃，但是该方法很容易出现丢失典型样本的问题；2、基于聚类的欠采样，即对多数类样本进行聚类，以聚类中心代替原始多数类样本。

但是，无论是过采样策略还是欠采样策略，二者均只能在一定程度上降低非均衡性带来的影响，但无法完全解决该问题，二者均有自己的局限性：过采样策略会导致模型容易出现过拟合现象；欠采样策略存在丢失原始数据信息的风险。

2.1.2 算法层面

除了在数据层面进行改进，研究者们也开始关注算法本身对非均衡问题的处理能力，比较典型的有两类算法：

1) 集成学习

集成学习，尤其是 Boosting 类算法具有两个特点很适合处理非均衡分类问题：1、数据重构，模型会根据上一轮迭代结果对训练数据进行重构，这种重构处理会使得模型更关注少数类；2、多个弱分类器的结合，每个弱分类器的训练样本不同，侧重点不同，往往精度也不同，通过多个弱分类器的结合能够极大地保证模型效果的稳定性。

2) 代价敏感学习

代价敏感学习的特点是打破不同类错分代价相同的假设，给予少数类错分更高的代价，降低多数类错分代价。基于代价敏感学习的原则，对传统的分类算法进行改造。主要方法有两个：1、提高少数类错分的代价函数，主要是通过提高权重实现；2、引入代价敏感矩阵。

2.1.3 评价指标层面

在评价模型分类结果时，需要选取合适的评价指标，传统对分类结果的评价方法都是从整体出发，即考虑分类结果整体的准确率。但非均衡分类问题往往会出现在少数类上的精度差，使用整体准确率评价非均衡分类问题往往会显示出不足。研究者们因此提出了一些更复杂，更合适的评价指标，例如：AUC 值、G-Mean、BAC 等[9]。

2.2 集成学习

集成学习作为一种强大的机器学习框架，不仅有很成熟的数学理论做支撑，而且在深度学习崛起之前在众多领域均取得了很好的效果[10]，其核心思想是通过某种策略构建并结合多个弱学习器，基于群体决策来提高整体决策的泛化能力。

集成学习的过程是先产生一组弱学习器，再用某种策略将它们结合起来。根据弱学习器的生成方式，目前的集成学习方法大致可分为两大类[9]173：1、串行生成弱分类器，代表方法为Boosting；2、并行生成弱分类器，代表方法包括 Bagging

2.2.1 针对分均衡分类的 Boosting 算法

Boosting 算法的一个优点是在迭代中对样本数据进行重构，并在此基础上进行学习，该特点使得Boosting 算法很适合非均衡分类问题，因此许多学者结合针对非均衡分类问题的处理策略对Boosting 算法进行了改进，这些改进方向大致可分为三类：

1) 采样方法+Boosting

该策略主要是将采样方法加入 Boosting 算法的每一轮迭代过程中。根据采样方法的不同，大致又可分为三类：1、随机采样：如 Seiffert 等人[11]在 Adaboost 基础上，对多数类进行随机下采样，提出了 RusBoost； Chawla 等人[12]对少数类进行SMOTE 过采样，从而形成了SMOTEBoost；2、基于数据增强的采样：如 Herna L Viktor 等人[13]在数据增强的基础上改进AdaBoost.m1，从而形成了 DataBoost-IM；3、基于聚类的采样，如 Rayhan 等人[14]在AdaBoost.m2 的基础上，对数据进行聚类下采样，从而形成了CUSBoost。

2) 代价敏感学习+Boosting

该策略主要是在Boosting 算法的基础上提高少数类错分代价，降低多数类错分代价。如 W Fan 等[15]从数据预处理角度改进，将代价用于数据权重调整的AdaCost 算法；Ting KaiMing[16]将代价敏感矩阵作为某种“先验”引入AdaBoost，提出了和MetaCost 相结合的CBS 算法。

3) Bagging + Boosting

该策略是将采样方法引入Boosting 算法，之后通过Bagging 的方式将多个Boosting 结合起来，进行决策，比较典型的算法包括 Ying Liu X 等[17]提出的EasyEnsemble 和BalanceCascade。

2.2.2 本文选取的 4 种基准模型

为了更好地评价深度Boosting 决策树算法在非均衡分类问题上的性能，本文选取了 4 类典型的Boosting 算法与其进行对比，包括：AdaBoost、XGBoost、SMOTEBoost 和RUSBoost。

1) AdaBoost

Freund 和 Schapire[18]提出的AdaBoost 作为一种最具有代表性的Boosting 算法，在很多场景都有成功的应用，这种方法具有一定的处理非均衡分类问题的能力，因此本文选择其作为一个比较模型。

2) XGBoost

XGBoost 是 Tianqi Chen 等[19]在 Friedman 等人[20]提出的 GBDT 算法基础上改进形成的一种更高效的Boosting 算法，该算法具有利用二阶导信息，并行计算等优点，因此，

本文选择 XGBoost 算法作为第二个比较模型。

3) SMOTEBoost

SMOTEBoost 是针对分均衡分类问题的一种基于AdaBoost.m2 的改进算法，特点是每次迭代前对少数类做SMOTE 过采样。

4) RUSBoost

RUSBoost 是针对分均衡分类问题的另一种基于AdaBoost.m2 的改进算法，特点是迭代前对多数类进行随机欠采样。

这四个 Boosting 算法可以分为两类：AdaBoost 和 XGBoost 并非是针对非均衡分类提出的，但二者均具有一定的能力；SMOTEBoost 和 RUSBoost 是在 AdaBoost 的基础上结合采样技术针对非均衡分类问题提出的改进算法。

2.3 软决策树

传统的集成学习的弱学习器通常会选择神经网络或者决策树。神经网络往往具有很好地泛化性能，这种能力取决于在隐藏层中特征的分布式表示[21]，但是这种表示往往难以被人类所理解，尤其是在deep learning 中隐藏层很多，对其做解释尤其困难，因此这种模型的可解释性很差，往往被称为“黑箱”模型。决策树作为一种经典的机器学习分类算法，由于可以将其认为是 if-then 规则的集合[22]，其呈现树形结构，因此它的可解释性很好，但是其泛化性能远远不及深度网络，此外在面对小样本数据时往往会出现欠拟合现象[23]。

因此，一个新的研究思路就是将两种模型进行融合，构造出一个软决策树模型，避免在泛化性和可解释性之间权衡。许多学者在这方面进行了尝试，如 2016 年 Kontschieder P 等人[24]提出了一种尝试将决策树与神经网络结合并进行端到端训练的方法，将随机森林作为深度神经网络最后一层的分类器，并解决了反向传播的问题；2017 年 Hinton 等人[23]使用神经网络蒸馏成功将一个神经网络进行蒸馏成了决策树，成功地将其泛化能力进行了迁移。

3 模型建立

3.1 构造软决策树

3.1.1 构造节点模型-神经网络

本文构造了一种软二决策树作为弱分类器，这种决策树中每一个节点d_i是一个神经网络模型，本文将决策树的节点分为两类：1、子节点；2、叶节点(决策树最后一层)，本文分别对子节点和叶节点建立模型。

1) 子节点

本文以单层神经网络为例[①]，子节点的输入空间为$\mathcal{X}=\mathbb{R}^p$，输出空间为$\mathcal{Y}=[0,1]$。模型是神经网络$d_i(\boldsymbol{x};\Theta)$，学习的参数$\Theta$包括权重向量$\boldsymbol{w}_i$和偏置$b_i$，节点网络的输入维度为$p$，本文设置输出维度为 1，对应选择右侧子路径概率。因此神经网络隐藏层的激活函数可以根据需要选择[②]，输出前一层的激活函数选择 Sigmoid 型激活函数中的 Logistic 函数。每一个子节点，选择右侧子路径的概率为

$$p_i=\sigma(\boldsymbol{w}_i^T\boldsymbol{x}+b_i) \tag{3-1}$$

式中：$\boldsymbol{x}\in\mathcal{X}$-模型输入；$\sigma$-Logistic 激活函数$\sigma(x)=(1+\exp(x))^{-1}$

我们取阈值为0.5，比较p_i与阈值的大小，据此判断路径选择，当$p_i\geqslant 0.5$时，选择右侧路径，当$p_i<0.5$时，选择左侧路径。本文分别以$\mathbb{I}_i^{\ell}$和$\mathbb{I}_i^{r}$表示节点i是否选择左侧路径和是否选择右侧路径，二者满足$\mathbb{I}_i^{\ell}+\mathbb{I}_i^{r}=1$。

$$\mathbb{I}_i^{\ell}=\begin{cases}1 & p_i<0.5\\ 0 & p_i\geqslant 0.5\end{cases} \tag{3-2}$$

2) 叶节点

与子节点不同，叶节点的输出维度是类别数k（对于二分类问题$k=2$），此外，每一个叶节点的输出要经过Softmax 函数，最终得到一个关于类别的概率分布Q^{ℓ}，叶节点ℓ输出类别k的概率值为

$$Q_k^{\ell}=\frac{\exp(\phi_k^{\ell})}{\sum_{k'}\exp(\phi_{k'}^{\ell})} \tag{3-3}$$

式中：ϕ^{ℓ}-第ℓ个叶节点的参数。

[①] 可以构造多层的神经网络，相应的选择右侧路径的概率为$p_i=\sigma(f(w_i^Tx+b_i))$

[②] 常用的激活函数包括：Logistic 函数、ReLU 函数、Tanh 函数等

3.1.2 软决策树的决策过程

本文构建的是软决策树，每个子节点都有两个子路径，因此树的节点数为2^d-1[③]，d为树的深度，这样就构成了一个层次化的决策树，如图 3-1 所示。

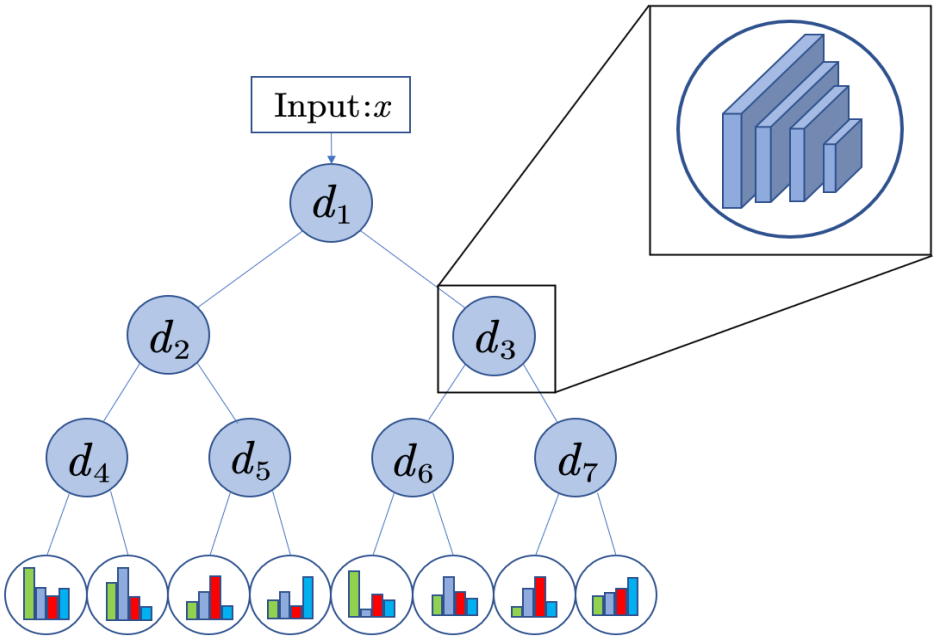

图 3-1　包含 7 个子节点 8 个叶节点 4 分类的多层网络软决策树示意图

1) 路径概率

考虑软决策树的一个具体的决策过程，假设输入一个样本$\boldsymbol{x}$，软决策树会根据前面提到的选择子路径过程进行决策，从而形成一条具体的决策路径，本文定义$\pi_i(\boldsymbol{x}|\Theta)$表示输入样本$\boldsymbol{x}$到达叶节点$i$的概率

$$\pi_i(\boldsymbol{x}|\Theta)=\sum_{1\le j<i} d_j(\boldsymbol{x};\Theta)^{\mathbb{I}_j^r}(1-d_j(\boldsymbol{x};\Theta))^{\mathbb{I}_j^\ell} \tag{3-4}$$

式中：$d_j(\boldsymbol{x};\Theta)$-子节点$j$选择右侧子路径的概率$p_j$。

对于到达叶节点的路径概率，很显然满足$\sum_{\ell\in Leaf\ Nodes}\pi_\ell(\boldsymbol{x}|\Theta)=1$。

2) 预测结果

软决策树的每个叶节点都会输出一个关于类别概率的分布，软决策树最终的决策结果是以路径概率为权重对叶节点输出分布的加权平均。因此对于样本$\boldsymbol{x}$软决策树的预测为类别k的概率为

$$\mathbb{P}[y=k|\boldsymbol{x},\Theta]=\sum_{\ell} Q_k^\ell \pi_\ell(\boldsymbol{x}|\Theta) \tag{3-5}$$

[③]第i层决策树的节点数为2^{i-1}，因此一个深度为d的决策树的所有节点数为$1+2+4+\cdots+2^{d-1}=2^d-1$

式中：Q_k^{ℓ}-叶节点ℓ输出类别k的概率。

3.1.3 软决策树的损失函数

对于软决策树的损失函数本文借鉴 Hinton 的做法[23]3，选取对数交叉熵函数，即取以路径概率加权的叶节点概率输出与目标分布之间的交叉熵，并取对数，如(3-6)所示

$$L(\boldsymbol{x}) = -\log\left(\sum_k T_k \log \mathbb{P}[y=k|\boldsymbol{x},\Theta]\right) \tag{3-6}$$

式中：T_k-样本$\boldsymbol{x}$的真实分布。

3.2 软决策树与 Boosting 框架结合

3.2.1 Boosting 框架

Boosting 算法作为一类集成学习算法具有很好的性质，本文总结 Boosting 算法的共同特征，形成一个关于 Boosting 算法的框架。Boosting 框架的两个特征是：

1、根据上一轮迭代弱学习器的结果重构训练数据，在新的数据分布下进行下一轮迭代；

2、采用某种策略若干弱学习器的结果结合，形成一个强学习器，如图 3-2 所示。

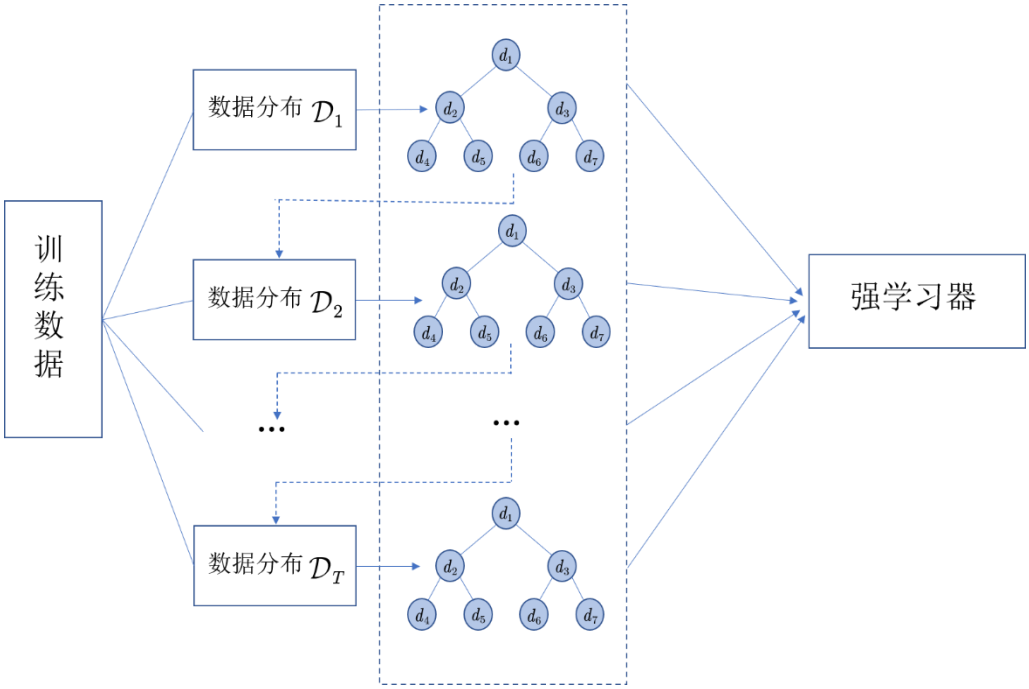

图 3-2　Boosting 框架示意图

3.2.2 软决策树+Boosting 框架

本文将软决策树作为弱分类器，并与 Boosting 框架进行结合，从而形成深度 Boosting 决策树算法。要实现二者的结合，有两个关键问题：1、如何重构训练数据；2、如何选择合适的结合策略。

在本文中，我们构建的是二分类模型，我们借鉴了针对二分类的离散 AdaBoost 算法[④]中的解决方法。

1) 数据重构

以本文为例，构建的弱分类器为软决策树模型$h_m(x):\mathcal{X}\to\{-1,+1\}$。该弱分类器是在分布为$\mathcal{D}_m$的重构数据上训练得到的，我们定义模型的分类错误率

$$\epsilon_m=\mathbb{E}_{x\sim\mathcal{D}_m}[h_m(x)\neq y] \tag{3-7}$$

式中：y-样本x的真实标签。

根据弱分类器的错误率可以赋予弱分类器相应的权重，错误率低的相应权重较高

$$\alpha_m=\frac{1}{2}\log\frac{1-\epsilon_m}{\epsilon_m} \tag{3-8}$$

根据上一轮模型的错误率我们对下一轮训练数据进行重构，降低上一轮中分类正确的样本的权重，提高分类错误的样本的权重。

$$\mathcal{D}_{m+1}(x)=\frac{\mathcal{D}_m(x)\exp(-\alpha_m h_m(x)y)}{Z_m} \tag{3-9}$$

2) 结合策略

集成学习常用的结合策略包括：平均法、投票法和学习法。本文采用加权平均的方法，权重由(3-8)确定，即我们的目的是训练一个具有T个弱分类器的加法模型：

$$f(x)=\sum_{t=1}^{T}\alpha_t h_t(x) \tag{3-10}$$

式中：T-软决策树个数，$h_t(x)$-第 t 个软决策树，α_t-该软决策树的决策权重。

3.3 模型的学习过程

3.3.1 含正则化项的目标函数

为了获得最终模型，需要构造合适的目标函数，并通过将其极小化学的模型参数，为了防止模型过拟合，本文构造了一个含有两个正则化项的目标函数，如式(3-11)。

$$\mathcal{L}=\sum_{i=1}^{N}l(f(x_i),y_i)+\sum_{t=1}^{T}(C_t+\Omega_t(w)) \tag{3-11}$$

式中：l-指数损失函数；C_t-软决策树t的正则化项 1；$\Omega_t(w)$-软决策树t的正则化项 2。

损失函数l是指数损失函数。

$$l(f(x_i),y_i)=y_i\exp(-y_i f(x_i)) \tag{3-12}$$

1) 正则化项 1

软决策树在训练过程中容易由于样本数据多样性不够导致子节点陷入对某一个特定的子路径的“依赖”[23]4, 从而造成欠拟合，因此 Hinton 在构造单层软决策树时加入

[④] AdaBoost 算法分类针对二分类的离散 Adaboost、针对多分类的离散 AdaBoost.m1 和针对多分类的连续 AdaBoost.m2，针对具体的分类任务，可以借鉴不同的 AdaBoost 算法对深度 Boosting 决策树算法进行改造。

了惩罚项为两条子路径选择的期望分布 0.5，0.5 和真实分布$\alpha_i,(1-\alpha_i)$的交叉熵。

式(3-4)代表了路径概率与节点输出的关系，用路径概率表示子节点d_i对其右侧子路径的选择概率，得到软决策树t关于重构数据$\mathcal{D}_t$的真实分布α_t

$$\alpha_i=\frac{\sum_{\boldsymbol{x}\sim\mathcal{D}_t}\pi_i(\boldsymbol{x}|\Theta)d_i(\boldsymbol{x};\Theta)}{\sum_{\boldsymbol{x}\sim\mathcal{D}_t}\pi_i(\boldsymbol{x}|\Theta)} \tag{3-13}$$

式中：$\pi_i(\boldsymbol{x}|\Theta)$为根节点到子节点$d_i$的路径概率

考虑到输入样本数据落在某一节点的数量随着决策树的深度d成指数衰减，本文在该惩罚项前加入了衰减系数2^{-d}。因此，决策树t的第一项完整的正则项为

$$C_t=-\lambda_1\times 2^{-d}\sum_{i\in Inner\ Nodes}0.5\log(\alpha_i)+0.5\log(1-\alpha_i) \tag{3-14}$$

式中：λ_1-模型超参数

2) 正则化项 2

为了防止过拟合，我们同时构造了正则项 2，即子节点内神经网络的权重向量的ℓ_2范数，软决策树t的第二项正则化项为

$$\Omega_t=\lambda_2\sum_{i\in Inner\ Nodes}|w|_2 \tag{3-15}$$

式中：λ_2-模型超参数；w-子节点内神经网络的权值向量。

3.3.2 前向分步算法视角

假设给定一个具有N个样本，p个特征的训练数据集$\mathcal{T}=\{(\boldsymbol{x}_i,y_i)\}$$(|\mathcal{T}|=N,\boldsymbol{x}_i\in\mathbb{R}^p,y_i\in\{-1,+1\})$。在给定目标函数比较复杂的情况下，极小目标函数是很困难的，因此我们从前向分布算法的角度出发，对目标函数进行分步优化。

第t步的优化目标函数为

$$(\alpha_t,h_t)=\arg\min_{\alpha,h}\sum_{i=1}^{N}L\left(y_i,f_{t-1}(x_i)+\alpha_t h_t(x_i)\right)+C_t+\Omega_t(w) \tag{3-16}$$

式中：L-关于指数损失函数的一种变形。

$$\begin{aligned}L(y,f_{t-1}(x)+\phi_t h_t(x))&=[y-f_{t-1}(x)-\phi_t h_t(x)]^2\\&=[r-\phi_t h_t(x)]^2\end{aligned} \tag{3-17}$$

式中：r-样本真实标签y与$f_{t-1}(x)$之间的残差，用$r=y\exp(-f_{t-1}(x)y)$表示。

得到参数α_t和软决策树h_t，更新$f_t(x)=f_{t-1}(x)+\alpha_t h_t(x)$

得到最终的决策模型为：

$$H(x)=\arg\max_{y\in\{-1,+1\}}P(f(x)=y|x) \tag{3-18}$$

式(3-18)也可以用符号函数来表示[5]，并且可以证明其达到了贝叶斯最优。

3.4 学习过程的总结

总结上述推导，深度 Boosting 决策树算法的学习过程可以大致分为三个过程：

1、根据重构数据训练软决策树的参数Θ，此过程本文使用小批量梯度下降；

2、根据迭代结果对训练样本进行重构；

3、将弱分类器进行结合。

整个算法的详细学习过程如算法 1 所示。

算法 1：深度 Boosting 决策树算法的学习过程

输入：训练集$\mathcal{T}=\{(\boldsymbol{x}_1,y_1),\cdots,(\boldsymbol{x}_N,y_N)\},\boldsymbol{x}_i\in\mathcal{X}=\mathbb{R}^p,y_i\in\mathcal{Y}=\{-1,+1\}$;模型超参数：软决策树数量$T$、树的深度$d$、树节点内网络层数$c$、正则项系数$\lambda_1,\lambda_2$和迭代次数$\mathrm{nEpochs}$

初始化数据权重$D_i(\boldsymbol{x})\leftarrow\frac{1}{N}$ for $i=1,\cdots,N$
for $t=1\rightarrow T$ **do**
    随机初始化[6]Θ
    for $i=1\rightarrow\mathrm{nEpochs}$ **do**
        将$\mathcal{T}$拆分成小批量
        for all minibatch from $\mathcal{T}$ **do**
            通过SGD更新Θ
        end for
    end for
    输出软决策树h_t
    $\epsilon_t\leftarrow\mathbb{E}_{x\sim\mathcal{D}_t}[h_t(x)\neq y]$
    $\alpha_t\leftarrow\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$
    $\mathcal{D}_{t+1}(x)\leftarrow\mathcal{D}_t(x)\exp(-\alpha_t h_t(x)y)$
    $\mathcal{D}_{t+1}(\boldsymbol{x})\leftarrow\frac{\mathcal{D}_{t+1}(\boldsymbol{x})}{\sum_{\boldsymbol{x}}\mathcal{D}_{t+1}(\boldsymbol{x})}$
end for

输出：预测模型$H(x)=\arg\max_{y\in\{-1,+1\}}P(f(x)=y|x)$

[5] $\underset{y\in\{-1,+1\}}{\mathrm{argmax}}P(f(x)=y|x)=\begin{cases}1,H(x)\geq 0\\-1.H(x)<0\end{cases}=\mathrm{sign}(H(x))$

[6] 本文采用 Xavier 初始化方法。

4　实验

本文选取了三个真实欺诈交易数据集用来做实验，涉及的业务场景包括：盗刷信用卡、还贷和存款。为了评价模型的性能，本文选取了三类评价指标，包括 $Accuracy$、AUC 和 $Precision$ $Recall$ $F1$。同时为了比较深度 Boosting 决策树算法的性能，本文选取四类Boosting 算法与其进行对比，包括：AdaBoost、XGBoost、SMOTEBoost 和 RUSBoost。本文构建的模型均为非均衡二分类模型，实验将在 Python3.7.10 平台上完成，框架使用 PyTroch.1.7.1。

4.1 实验数据

4.1.1 数据集介绍

具体来讲，这三个数据集包含的数据均为非均衡分类数据，其中包含若干个特征，目标分为正常和异常两种类型，绝大部分数据为正常类型，少部分属于异常类型，即欺诈交易，属于典型的非均衡二分类问题。

1）数据集 1

该数据集来源于 Worldline 和 ULB 大学的机器学习小组合作收集的欺诈检测数据集[⑦]，包含 2013 年 9 月欧洲持卡人两天内通过信用卡进行的交易。本文选取部分的数据集包含 25135 笔交易，其中有 422 起欺诈交易。该数据集包含 17 个特征，目标是该笔交易是否为盗刷信用卡行为。

2）数据集 2

该数据集来源于一家美国金融互联网公司 Lending Club，包含 2007 年至 2015 年的历史交易数据，本文从 Kaggle 平台上获取了数据[⑧]，并选取了其中部分数据，包含 95791 笔贷款交易，其中 153 笔贷款交易违约，包含 13 个特征，目标是该笔贷款的是否会发生违约现象。

3）数据集 3

该数据集来源于 UCI 大学开源数据集中的 Bank Marketing 数据集[⑨]，包含了西班牙银行机构关于定期存款产品的订阅情况(虽然不是严格的欺诈交易数据，但属于真实交易场景下非均衡分类数据，因此可用于评估模型性能)。该数据集包含 41189 笔交易数据，订阅交易 4641 笔，包含 18 个特征，目标是用户是否会订阅。

4.1.2 数据集统计情况

这三个数据集均包含两类数据，正常类型和欺诈类型，两类样本的数据量相差悬殊，对于这三个数据集的基本统计情况，包括样本数、输入数据特征数、类别数以及欺诈类

⑦ https://mlg.ulb.ac.be/wordpress/projects/
⑧ https://www.kaggle.com/swetashetye/lending-club-loan-data-imbalance-dataset
⑨ https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

型占样本总体的比例，如表 4-1 所示。

表 4-1 数据集基本统计情况

数据集	样本数	特征数	类别数	少数类占比(%)
Credit Card	26107	17	2	5.3
Loan Data	95791	13	2	1.6
Bank Market	41189	17	2	11.2

此外，每个数据集包含众多的特征，表 4-2 展示了每个数据集包含的特征。

表 4-2 数据集特征描述

Credit Card	Loan Data	Bank Market
gender	*credit.policy*	*age*
age	*purpose*	*job*
education	*interest.rate*	*marrage*
car	*installment*	*education*
reality	*annual.income*	*house*
child	*debt.to.income*	*loan*
income	*fico*	*default*
job	*days.cr.line*	*contact*
years.employed	*recol.balance*	*duration*
family.type	*revol.bal.rate*	*campaign*
family.size	*inqu.last 6mths*	*pdays*
house.type	*inq.2yrs*	*previous*
flag.mobil	*public.record*	*poutcome*
phone	-	*emp.var.rate*
work.phone	-	*cons.price.idx*
email	-	-
begin.month	-	-

4.1.3 数据预处理

这三个数据集的特征可以分为两类：类别特征和数值型特征，类别特征又包含有序和无序两类。因此在使用这些数据之前需要进行数据预处理，主要是三方面：1、one-hot 编码处理无序类别特征；2、顺序编码处理有序类别特征；3、对数值型特征归一化。

1) one-hot 编码

上述三个数据集均含有类别特征，有一些类别特征中的不同类别之间是无序的，例如数据集 Credit Card 中的性别特征[“男”，“女”]、数据集 Loan Data 中的贷款目的特征[“信用卡”，“债务合并”，“教育”，“家庭装修”，“小生意”，“其他”] 和数据集 Bank Market 中的用户婚姻状况特征[“单身”，“结婚”，“离异”，“再婚”]。

由于计算机只能接受数字输入，因此需要对原始类别特征进行数字编码，在处理这些不含顺序信息的类别特征时，无法使用含顺序信息的编码方式，否则就会加入原始数据不包含的信息。因此，本文采用 one-hot 编码，具体来说是将类别特征映射成一

个 one-hot 向量。以数据集 Credit Card 中的性别特征[“男”，“女”]为例，该类别特征包含 2 类，每一类对应一个长度为 2 的 one-hot 向量，即“男”=[1,0]“女”=[0,1]。

2) 顺序编码

同样，三个数据集中也包含一些类别之间有顺序的类别特征，如数据集 Credit Card 中的教育背景特征[“Lower secondary”，“Secondary / secondary special”，“Incomplete higher”，“Higher education”，“Academic degree”]。对于这类特征，由于不同类之间存在顺序信息，因此无法采用 one-hot 编码方式。

本文采用常用的顺序编码，即将类别特征映射成不同的自然数，以教育背景特征为例，按照顺序将其映射成“Lower secondary=1，“Secondary / secondary special”=2，“Incomplete higher”=3，“Higher education”=4，“Academic degree”=5。

3) 归一化

除了类别特征，这三个数据集中同样存在许多数值型特征，如数据集 Credit Card 中的工作年限特征、收入特征、数据集 Loan Data 中的贷款利率特征和数据集 Bank Market 中的用户的年龄特征。这些特征都是数值型特征，因为模型采用的是欧氏距离，因此如果直接输入模型会造成由于不同类别量纲的不同发生某些类信息被掩盖的现象。因此，我们对数值型特征进行归一化。

4.2 评价指标

4.2.1 准确率与错误率

为了准确地评价模型的性能需要选取合适的评价指标。对分类模型最常用的评价指标是准确率$Accuracy$和错误率$error\ rate$，这两个指标虽然在面对非均衡分类问题具有一些局限，但由于可以从这两个指标上看出模型的整体性能，因此我们选择其作为第一类评价指标。

$$Accuracy = \mathbb{E}[I(f(x) = y)] \tag{4-1}$$

4.2.2 查准率、查全率与 F1

考虑到常用评价指标$Accuracy$和$error\ rate$在面对非均衡分类问题时会出现多数类信息掩盖少数类信息的现象[9]30，并且由于本文仅考虑了二分类问题，因此我们引入了混淆矩阵，如表 4-3 所示，并基于此计算得到“查准率”($Precision$)和“查全率”($Recall$)

表 4-3 混淆矩阵

	预测正例	预测反例
真实正例	真正例(TP)	假负例(FN)
真实负例	假正例(FP)	真负例(TN)

查准率$Precision = TP/(TP + FP)$，该指标关心模型在正类中真正识别出的数量，反映的是模型识别类的精度；查全率$Recall = TP/(TP + FN)$，该指标关心在模型识别为少数类真正有多少正类，反映的是模型识别类的能力。二者经常是一对矛盾，无法兼得，因此本文又选取了二者的调和平均数：$F1$。

$$F1 = \frac{2 \times Precision \times Recall}{Recall + Precision} \tag{4-2}$$

4.2.3 ROC 曲线与 AUC 值

在分类问题中，我们学习到的分类器通常是输出一个预测概率，我们将这些概率值进行排序，在中间选取“截断点”，前半部分分为正例，后半部分分为负例。该方法对于阈值的选取过于主观，因此研究者们从医学领域引入Roc曲线。曲线横轴是“假正例率”$\mathrm{FPR} = FP/(TN + FP)$，纵轴是“真正例率”$\mathrm{TPR} = TP/(TP + FN)$，通过这种方式将结果映射到平面上一点，选择不同阈值最终获得一条Roc曲线，该曲线具有不受样本非均衡性影响的优势。

为了方便进行不同算法之间的比较，学者们定义AUC值代表曲线下方面积，通过比较AUC值来反映模型的性能，AUC值越大表示模型性能越好。

4.3 实验设计及结果

本文使用分层随机抽样的方法将原始数据集划分为训练集(70%)和测试集(30%)，训练集和测试集中少数类占比相同。同时为了保证结果的可靠性，本文使用K折交叉验证$(k = 5)$，多次训练模型进行预测，并对结果取平均值。

4.3.1 五种算法的比较

我们分别在 Credit Card、Loan Data 和 Bank Mark 数据集上训练五种模型，并在测试集上进行预测，在测试集上得到预测结果。

1) 在数据集 1 上结果比较

在数据集 Credit Card 上训练模型，并根据三类评价指标进行评价，结果如表 4-4 所示，并得到 ROC 曲线如图 4-1 所示。

表 4-4 在 Credit Card 数据集上结果比较

模型	指标 1	指标 2	指标 3		
	Accuracy	AUC	*Precision*	*Recall*	$F1$
1) AdaBoost	98.1%	0.54	100%	5%	0.09
2) XGBoost	98.4%	0.54	100%	5%	0.09
3) SMOTEBoost	93.5%	0.62	4%	13%	0.06
4) RUSBoost	89.1%	0.61	3%	54%	0.05
5) 深度 Boosting 决策树	98.3%	***0.68***	67%	5%	0.09

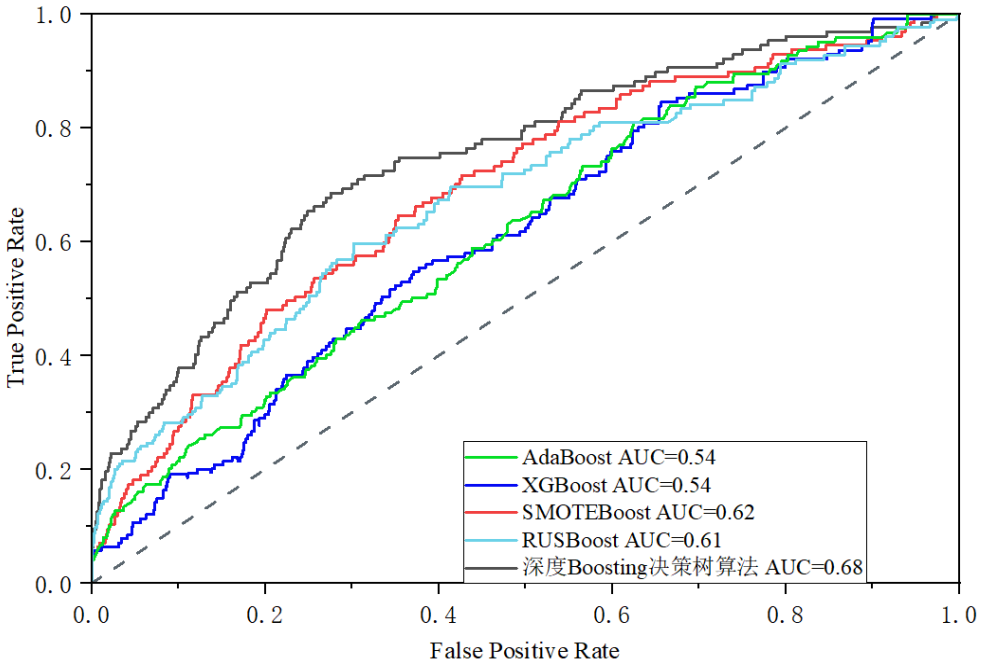

图 5-1　5 种算法在 Credit Card 数据集上的 ROC 曲线

从*Accuracy*指标来看，深度Boosting 决策树算法、AdaBoost 和XGBoost 算法性能接近，准确率能达到98% 以上，说明其整体性能还是比较好的，而针对非均衡分类问题改进的算法SMOTEBoost 和RUSBoost 性能比较接近，它们整体准确率反而相对低一些分别为93% 和89% 。

而从AUC 指标来看，深度Boosting 决策树算法远高于其他四类算法，AUC 值分别达到了0.68，AdaBoost 和XGBoost 算法的AUC 值接近，均为0.54 略好于随机猜测[10]，而改进算法SMOTEBoost 和RUSBoost 的AUC 值也接近，均大于0.60，说明其针对非均衡分类问题的改进措施取得效果。

从指标*Precision* 、*Recall* 、*F*1 来看，深度Boosting 决策树算法、AdaBoost 和XGBoost 算法比较接近，它们的*Precision* 指标更高一些，而*Recall* 相对较低，说明其对少数类分类的精度更高，但识别能力较弱。

从 ROC 曲线上判断，这 5 类算法的AUC 值均不是很高，按照AUC 进行排序可以将算法大致分为三类：1、深度Boosting 决策树算法（AUC 值最高）；2、改进算法SMOTEBoost 和RUSBoost（改进措施取得一定效果，AUC 值低于第一类，但高于其

[10] 随机猜测表示对样本随机分类，其 AUC 值为 0.5，在 ROC 曲线中体现为 0-1 之间的直线。

他类)；3、AdaBoost 和XGBoost 算法(针对非均衡分类效果最差，AUC 值最低)。

2) 在数据集 2 上结果比较

类似地，在数据集 Loan Data 上训练模型，并根据三类评价指标进行评价，结果如表 4-5 所示，并得到 ROC 曲线如图 4-2 所示。

表 4-5 在 Loan Data 数据集上结果比较

模型	指标 1	指标 2	指标 3		
	Accuracy	AUC	*Precision*	*Recall*	*F*1
1) AdaBoost	83.5%	0.51	43%	8%	0.14
2) XGBoost	83.8%	0.53	58%	2%	0.03
3) SMOTEBoost	81.0%	0.54	30%	14%	0.19
4) RUSBoost	79.0%	0.56	21%	46%	0.29
5)深度 Boosting 决策树	83.9%	***0.66***	69%	4%	0.08

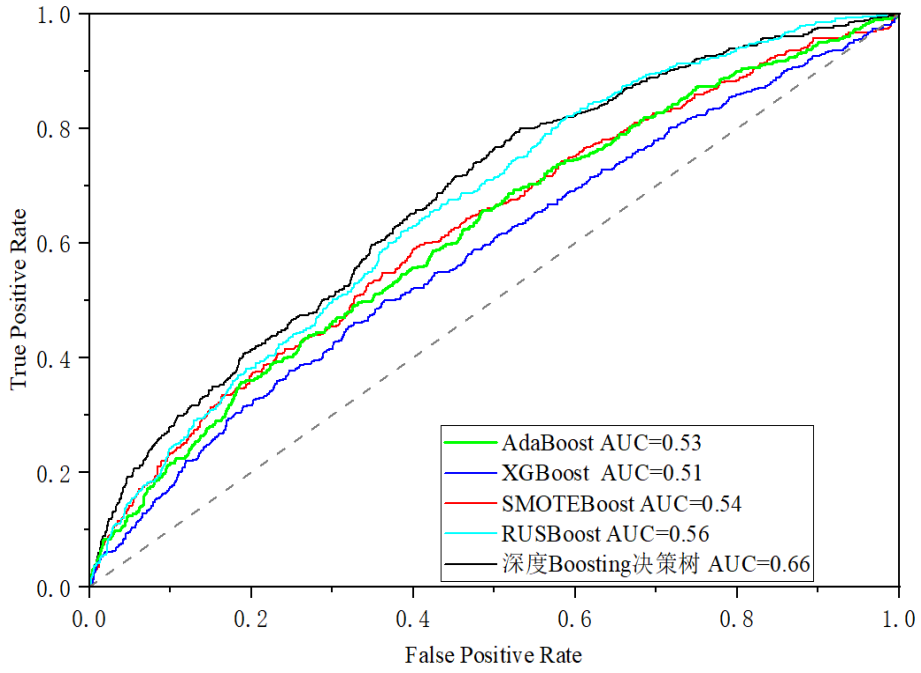

图 4-2　5 种算法在 Loan Data 数据集上的结果

类似地，我们从 *Accuracy* 指标来看，同样能发现深度 Boosting 决策树算法、AdaBoost 和XGBoost 算法性能接近，其准确率达到83% 高于算法SMOTEBoost 和RUSBoost 的81% 和79% 。

而从AUC 指标来看，深度Boosting 决策树算法远高于其他四类算法，AUC 值分别达到了0.66，其他四类算法的AUC 均未超过0.60 。

从指标 *Precision* 、*Recall* 、*F*1 来看，除了RUSBoost 呈现出和在 Credit Card 数据集上不同的特征，其他算法呈现的特点与之前保持一致。

3) 在数据集 3 上结果比较

类似地，在数据集 Bank Market 上训练模型，并根据三类评价指标进行评价，结果如表 4-6 所示，并得到 ROC 曲线如图 4-3 所示。

表 4-6 在 Bank Market 数据集上结果比较

模型	指标 1	指标 2	指标 3		
	Accuracy	AUC	*Precision*	*Recall*	*F*1
1) AdaBoost	90.8%	0.68	66%	38%	0.49
2) XGBoost	91.9%	0.75	68%	53%	0.60
3) SMOTEBoost	89.7%	0.82	53%	73%	0.53
4) RUSBoost	86.1%	0.87	44%	90%	0.59
5)深度 Boosting 决策树	90.9%	***0.88***	88%	77%	0.82

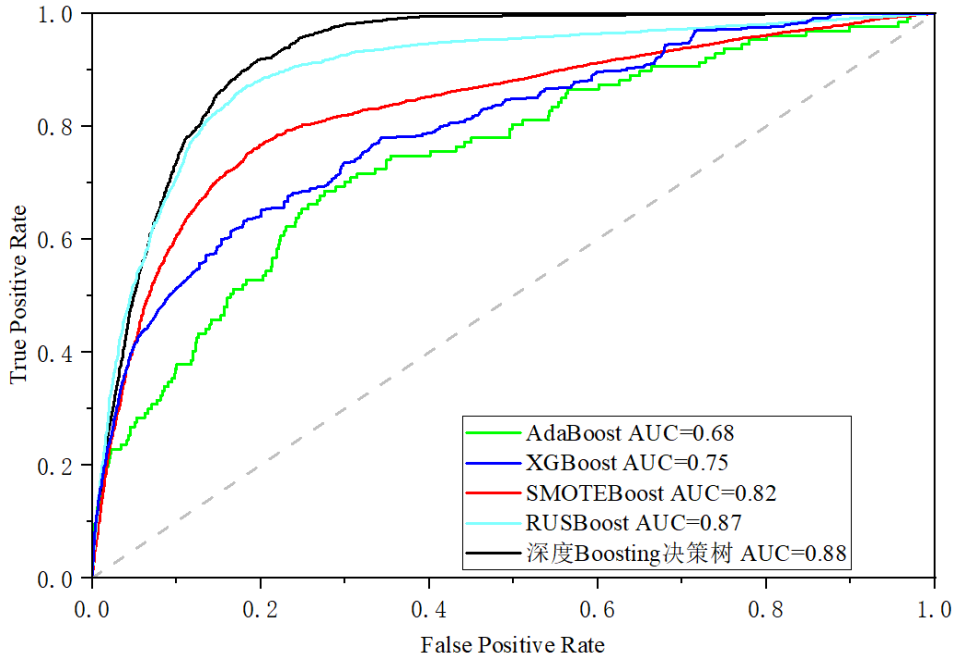

图 4-3　5 种算法在 Bank Market 数据集上的结果

在该数据集上，我们前面发现的特点体现地更明显。从 *Accuracy* 指标来看，深度 Boosting 决策树算法、AdaBoost 和 XGBoost 算法性能接近，准确率能达到 90% 以上，而 SMOTEBoost 和 RUSBoost 算法的准确率相对较低一些，二者性能比较接近。

而从 AUC 指标来看，深度 Boosting 决策树算法依然高于其他四类，AdaBoost 和 XGBoost 算法最低，但有趣的是 RUSBoost 的性能接近了深度 Boosting 决策树算法。

从指标 *Precision* 、 *Recall* 、 *F*1 来看算法的特性与在数据集 1 上的结果保持了一致。

4.3.2 Friedman 检验与 Nemenyi 检验

前面我们得到了 5 种算法在 3 个数据集上的结果，并根据指标对不同算法进行了排序比较，但是这种比较过于主观，因此，我们希望采用假设检验的方法，分析不同算法的差异是否显著。本文首先使用 Friedman 检验法检验算法的性能是否显著不同，接着使用 Nemenyi 检验进一步区分各算法。

1) Friedman 检验

首先本文根据AUC值将算法进行排序，性能最好的赋值 1，次之赋值 2，…, 若两个算法性能相近则取均值，排序结果如表 4-7 所示。

表 4-7 算法比较序值表

数据集	深度 Boosting 决策树	RUSBoost	SMOTEBoost	XGBoost	AdaBoost
Credit card	1	2.5	2.5	4.5	4.5
loan Data	1	2	3	5	4
Bank Market	1.5	1.5	3	4	5
平均序值	1.5	2.0	2.8	4.5	4.5

本文将采用 Friedman 检验变量进行假设检验，原假设H_0:“所有算法的性能相同”，统计量为

$$\tau_F = \frac{(N-1)\tau_{\chi^2}}{N(k-1)-\tau_{\chi^2}} \tag{4-3}$$

式中：N-数据集个数；k-算法个数；τ_{χ^2}-一个服从χ^2的统计量[11]。

该变量服从自由度为$k-1$和$(k-1)(N-1)$的F分布，将表 4-7 内的数据代入(4-3)，计算得到$\tau_F = 16.0$。查表得到，α为 0.05 和 0.1 时的临界值分别为 3.83 和 2.80。显然τ_F远大于临界值，因此在显著性水平α分别为 0.05 和 0.1 条件下拒绝“所有算法性能相同”的原假设。

2) Nemenyi 检验

我们在得到拒绝“所有算法性能相同”的原假设的结论后，下一步想探究的是这几种算法有哪几种是性能接近的，有哪些是性能显著有差异的。本文将借助 Nemenyi 检验，即查表得到在不同显著性水平α下的平均序值差别的临界值域

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{4-4}$$

式中：q_α-Turkey 分布的临界值。

查表得到，α分别为 0.05 和 0.1 条件下的q_α分别为 2.73 和 2.50，相应的临界值域分别为 3.5 和 3.1。与不同算法的平均序值之间的差进行比较，发现可以将算法分成两类：深度 Boosting 决策树算法、RUSBoost 算法和 SMOTEBoost 算法之间没有显著性差别，AdaBoost 算法和 XGBoost 算法之间没有显著性差别，但两类算法之间有显著性差别。

[11] $\tau_{X^2} = \frac{12N}{k(k+1)}(\sum_{i=1}^{k} r_i^2 - \frac{k(k+1)^2}{4})$，该统计量服从自由度为$k-1$的卡方分布

此处我们得到的结果与前面有一点不同，前面是将深度 Boosting 决策树算法单独划分为一类，原因是我们是根据AUC 值进行排序，此处仅考虑了排序信息，从而忽略了AUC 值之间的差值的大小。但尽管如此，我们仅使用排序信息就可以得到和前面相似的结果，通过假设检验方法验证了这五种算法之间的差异的显著性。

4.3.3 结论

对整个实验的结果进行总结，可以得到以下几个方面的结论：

1) 深度Boosting 决策树算法针对非均衡问题的分类效果整体优于其他四类算法。从三个数据集上的结果，我们可以看到无论是准确率还是AUC 值，深度Boosting 决策树都是高于其他算法，说明该算法确实对于解决非均衡分类问题具有优势。

2) 深度Boosting 决策树算法和SMOTEBoost RUSBoost 性能更接近；AdaBoost 和XGBoost 算法性能更接近。具体而言，第一类算法均考虑到非均衡分类问题，从数据重构的角度进行了改进，因此往往在少数类上的效果优于其他算法；第二类算法在多数类上的效果很好，因此往往显得整体准确率很高。

5 超参数对模型的影响

本文构建的深度Boosting决策树模型包含众多超参数，例如：软决策树的数量T、软决策树的深度d、子节点神经网络的层数c和惩罚项系数λ_1 λ_2。为了探究这些超参数对模型性能的影响，本文以Bank Market和Loan Data数据集为例，使用5－fold交叉验证，并对指标求平均值。

5.1 软决策树数量对模型性能影响

集成学习的基本思想就是集成若干个弱分类器形成一个强分类器，因此弱分类器的数量是一个很重要的超参数，为了探究软决策树的数量对模型性能的影响，本文基于Bank Market数据集构建一个树的深度d为2，神经网络层数c为1，惩罚项系数λ_1 λ_2分别为0.05和0.005的深度Boosting决策树模型，仅改变软决策树的数量，以10为步长，观测模型从包含10个决策树到100个决策树在测试集上的预测结果，这里指标选取了准确率和AUC值，训练轮次epoch会根据何时达到模型最好性能调整，结果如图5-1所示。

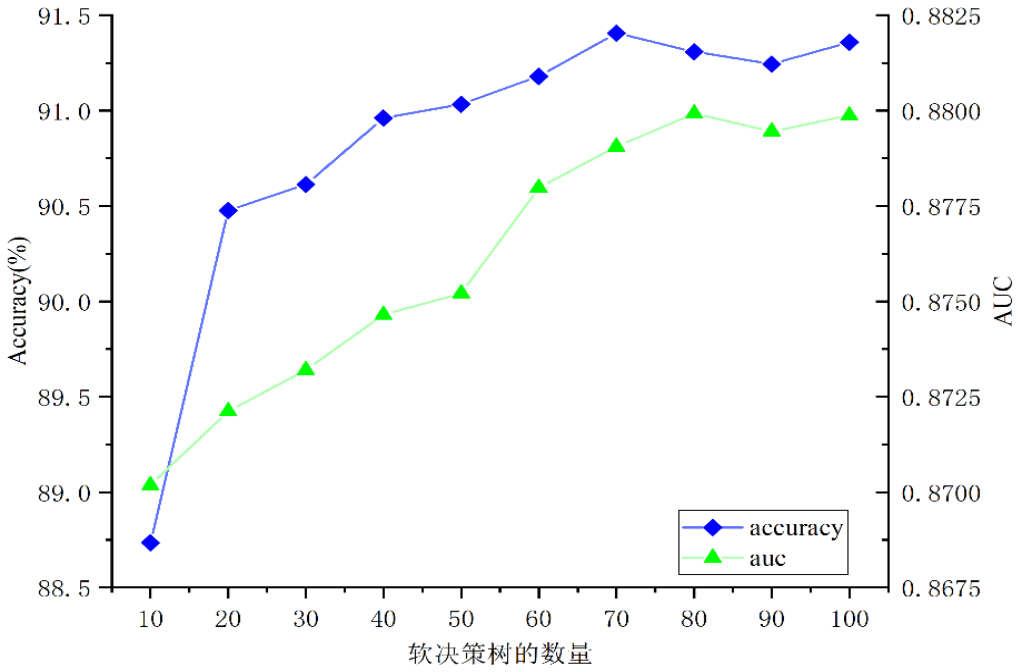

图 5-1　软决策树数量对深度 Boosting 决策树模型性能影响

可以看到，随着软决策树的数量增加，模型的准确率和AUC值均会有很大的变动，从实验的结果可以得到如下两个结论：

1)　增加软决策树的数量会提高模型性能，但存在性能的上限。从图5-1可以看

到，在软决策树数量增加到 70 左右时，模型基本上到达了性能的上限，再增加数量，提升的幅度很小，甚至会出现一定程度的下降。

2) 在调参过程中观测指标选择准确率或AUC 值均可。这两个指标表现出的趋势是相同的，因此对于非均衡分类模型的调参，可以选择简单指标-准确率。

5.2 软决策树的深度对模型性能的影响

除了软决策树的数量，决策树的深度d 也是模型的一个重要超参数，前面提到树的深度d 和树的节点数n 满足指数关系$n=2^d-1$，而每一个节点在软决策树中对应一个神经网络，如果选取的d 过大，将导致模型的复杂度远远大于实际需求，因此确定合适的树的深度极其重要。

我们设其他参数为：决策树数量T 为 100、神经网络隐藏层层数c 为 1、正则化项系数λ_1 λ_2 分别为 0.05 和 0.005。树的深度d 变化范围为 2-5，我们观察模型的准确率和AUC 值，结果如图 5-2 所示。

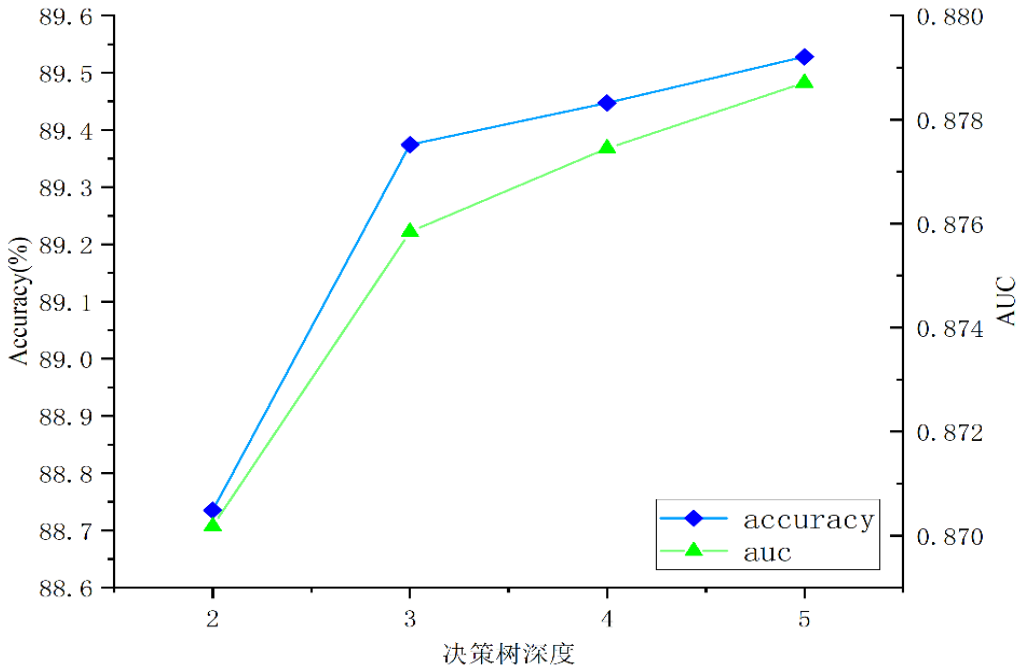

图 5-2　软决策树的深度对模型性能的影响

和软决策树的数量T 对模型性能的影响类似，在软决策树较浅时提升树的深度会提高模型的性能，但随着深度的增加这种提高会变缓。

这里可以得到如下两个结论：

1) 增加软决策树的深度会提高模型性能，但存在性能的上限。类似地、从图 5-2 可以看到，软决策树深度在较小时，会大幅提升模型性能，但是在软决策树达

到 4 和 5 左右就到达了上限。

2) 这里本文给出一个关于确定树深度的经验公式$d = \log_2 p$，其中p是输入数据的维度。之所以给出一个这样的公式，是因为我们提出软决策树的目的之一是借用决策树的结构粗略地表示模型的决策过程，在决策树中决策过程围绕特征分裂展开，因此我们希望软决策树的子节点数至少要保证每一个输入特征有机会进行“分裂”，因此希望软决策树的节点数要大概等于特征数。另外、图 5-2 的结果也符合该公式，输入特征数为 18，在树的深度为 4 左右，性能达到上限。

5.3 决策树数量和树的深度对模型影响的交互作用

我们单独分析了决策树数量和深度对模型性能的影响，我们接下来希望探究二者对模型性能是否具有交互作用。

本文保证其他超参数固定，同时变动决策树数量和树的深度，变动范围同单独分析时的范围相同，此时由前面结果可知AUC 值和准确率均可反映模型的性能，因此此处我们仅观察AUC 值，结果如图 5-3 所示。

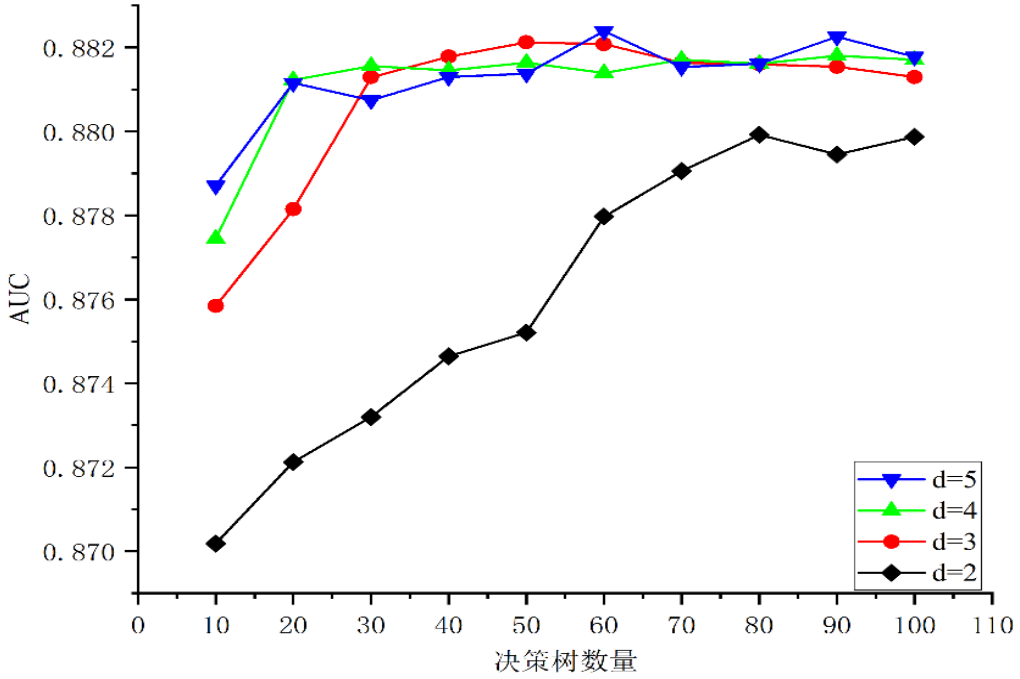

图 5-3　软决策树数量和深度对模型性能的影响

通过上图我们可以看到，单独增加软决策树的数量或树的深度均可以提高模型性能，并且随着树的深度的提高，要达到模型性能最优需要的决策树的数量会减少。从模型复杂度的角度，增加软决策树的数量或树的深度是提高模型复杂度的两种途径，而当模型复杂度超过具体的任务需求时，无法再提升模型性能。

5.4 神经网络层数对模型的影响

5.4.1 对性能的提升作用

为了探究神经网络层数对模型性能的影响，我们在Loan Data数据集上进行比较。超参数设置为：神经网络的激活函数我们选择Sigmoid函数、软决策树数量T为40、惩罚项系数λ_1 λ_2分别为0.1和0.005。我们比较了在树的深度分别在2和3两种条件下，模型在测试集上的预测准确率和AUC值的均值，结果如图5-4所示。

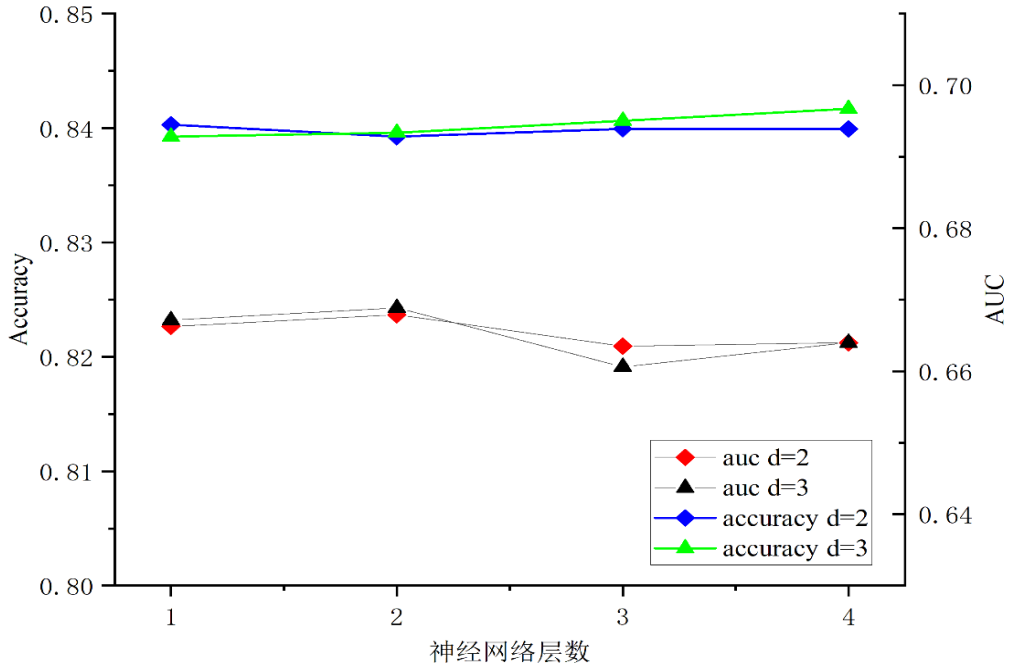

图5-4　神经网络层数对模型性能的影响

根据图5-4的结果，我们发现从预测准确率和AUC值两个指标来看，针对该任务，仅增加神经网络的层数模型的准确率和AUC值几乎没有差别，说明增加模型的层数并不会提高模型性能。

由于本文是利用结构化的数据训练模型，因此相对于该模型来说有可能该任务可能过于简单，使用单层的网络就已经能取得很好的效果，增加层数无法提高性能。但是对于更复杂的分类任务，如手写数字识别，Hinton等人[23]5构建的单层神经网络软决策树在测试集上的分类的正确率高达96.76%。

因此，本文认为仅仅增加神经网络的层数并不能提升该模型的性能。

5.4.2 层数过多带来的问题

在构建神经网络时，设置过多的隐藏层层数会增加模型复杂度，使得模型训练速度降低，同时更容易出现过拟合现象以及性能的不稳定。本文探究了在Loan Data数据集上进行非均衡分类时，增加深度 Boosting 决策树的神经网络层数是否会出现同样的

问题。

构建传统的神经网络时常常采用正则化来防止过拟合，本文在构建单层神经网络时引入的惩罚项 2 时采用了软决策树子节点的权重向量的ℓ_2范数，对于多层的神经网络本文将其拓展成每层权重向量的ℓ_2范数。我们分别在层数c为 1、2、3 和 4 时训练模型，并在训练轮次从 1 到 200 时观测其在测试集上表现。结果如图 5-5 所示。

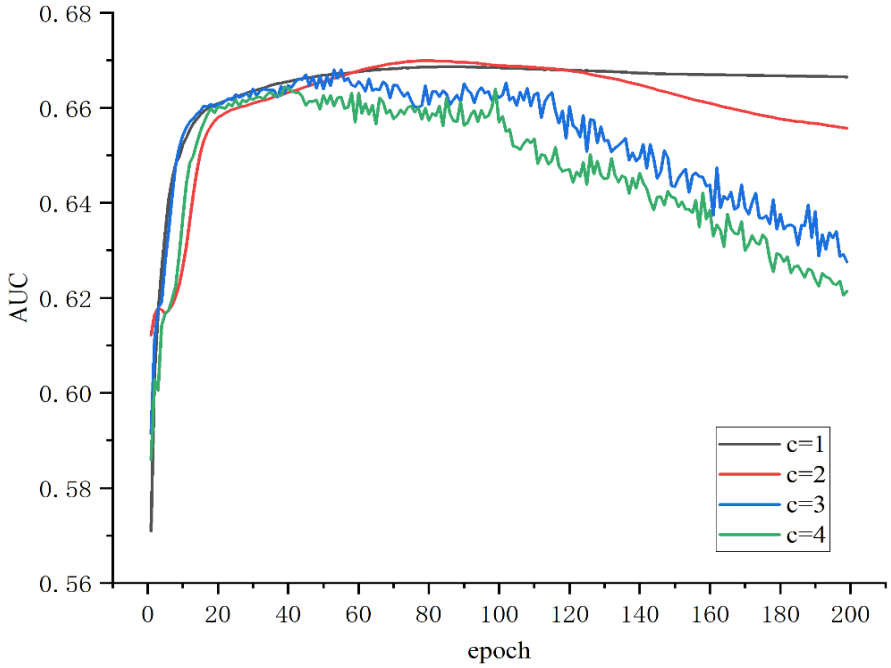

图 5-5　不同层数的训练过程

通过该实验，我们可以看到随着神经网络层数的增加，训练出来的模型反而会出现一些新的问题：

1）层数增加导致过拟合。我们可以看到，随着层数的增加，模型的AUC 值随着训练轮次的增加反而会下降，而且层数越多下降的幅度越大，这说明模型性能出现了下降，训练过程出现了过拟合的现象，并且层数越多这种倾向越明显，过拟合现象越严重。

2）层数过多，导致模型不稳定。我们观察到，在层数大于 2 时，模型的AUC 值开始出现波动，模型性能变得极不稳定，这是因为模型出现了过拟合，过于关注样本数据的“细节”，当模型过于关注上一轮迭代数据的某一些“细节”时在预测下一轮训练数据时就会出现预测错误。

5.5 结论

本文设置了一系列实验，探究了包括软决策树的数量T、软决策树的深度d、子节点神经网络的层数c等模型超参数对模型性能的影响。对这些结果进行总结得到了如下几个结论：

1）增加软决策树的数量能够提升模型性能，但存在性能上限。因此在确定该参数

时可以设置一个比较大的初始值。

2) 增加软决策树的深度也同样能够提升模型性能，初始确定树的深度可以参考输入数据维度的对数。
3) 软决策树的数量和深度均可以提升模型性能，并且树的深度越大达到模型性能上限所需要的树的数量越少。
4) 提升神经网络的层数，无法显著提升模型性能，反而会导致过拟合和模型性能不稳定的现象。

6 打开模型黑箱

以深度学习为代表的许多机器学习方法的一个被人诟病的一点是可解释性很差，因此被人称为是一个黑箱模型。而在许多具体的应用场景中，企业运营者对模型的期望不仅包括模型精度越高越好，而且希望模型的可解性很好，以欺诈检测为例，企业运营者不仅关心模型的准确率，更是希望模型能回答诸如“为什么这个人欺诈的可能性比其他人高？”这样的问题。因此本文希望兼顾泛化性能和可解释性，构建将神经网络和决策树融合的软决策树，并将其作为弱分类器与Boosting框架结合。我们尝试从特征重要性排序的角度打开模型。

6.1 特征重要性排序

本文在Loan Data数据集上进行预测，该任务是预测借款人是否会全额还贷，数据集包含借款人的14个特征，这些特征具体的含义如表6-1所示。

表6-1 Loan Data数据集的特征含义

特征名称	特征含义
credit.policy	1-满足公司借款标准；0-否
purpose	贷款目的：信用卡、债务合并、教育、家庭装修、小生意、其他
int.rate	贷款利率
installment	借款人分期付款金额
log.annual.inc	借款人年收入的自然对数
dti	借款人的债务-收入比率
fico	借款人的FICO信用评分
days.with.cr.line	借款人获得信用额度的天数
revol.bal	借款人的循环余额(信用卡账单周期结束时未付的金额)
revol.util	借款人的循环余额利用率(使用的信贷额度相对于可用信贷总额)
inq.last.6mths	借款人在过去6个月征信被查询的次数
delinq.2yrs	借款人在过去2年内逾期还款30天以上的次数
pub.rec	借款人的负面公共记录数量(破产申请、税收留置权或判决)
not.fully.paid	借款人是否全额还款

由于我们将神经网络嵌入决策树节点，因此无法使用决策树传统基于基尼系数和信息增益[9]75，78，79的方式计算特征重要性，我们利用OOB(Out of Bag)的方式计算特征重要性。具体来说，首先使用训练好的模型对OOB数据进行预测，计算出准确率或AUC等评价指标得分；接着对OOB数据中的每个特征进行逐一随机打散，重新计算得分，根据得分变化率进行特征重要性排序。

我们使用AUC指标作为打分标准，并对特征重要性进行排序，结果如图6-1所示。

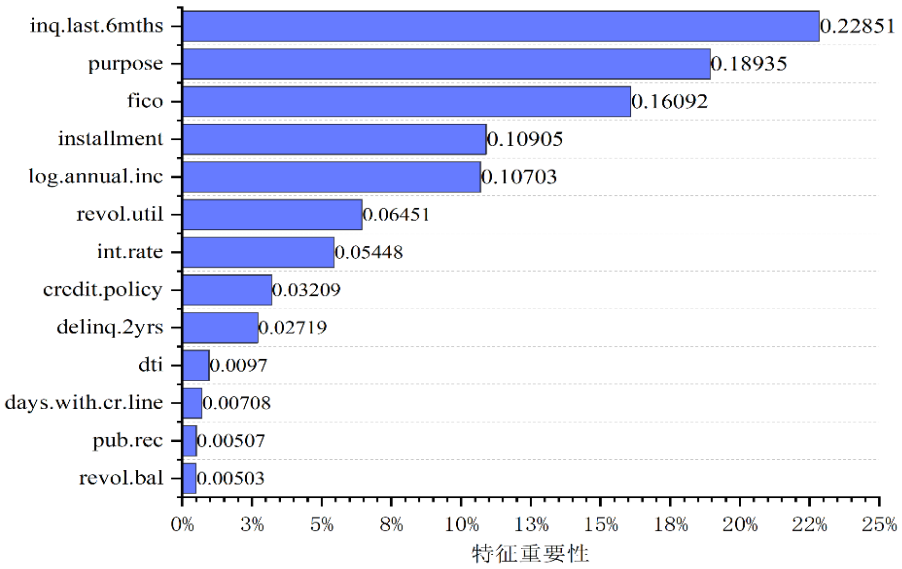

图 6-1 特征重要性排序

如图所示，我们发现特征重要性排序前 5 名分别为 inq.last.6mths(22.8%)、purpose(18.9%)、fico(16.1%)、installment(10.9%)和 log.annual.inc(10.7%)。

6.2 特征排序结果分析

排名第二的指标是借款目的(purpose)，后三个特征是借款人的个人信用和个人财产情况。不同类型的贷款的还款情况具有明显差异，而模型学习到了该差异，因此该指标在模型中的重要性很高，该结果证明了李思瑶等人[25]认为借款目的对违约风险无显著影响的观点不成立；个人信用好并且收入情况好的借款人还款的可能性比较大，这些指标的重要性高，这与人们的认知相符的。但是最重要的特征是个人征信查询次数(inq.last.6mths)，是否满足公司借款标准(credit.policy)的重要性并不像人们认知中的那样高，因此接下来我们将详细分析这两个指标。

6.2.1 个人征信查询次数

根据特征重要性排序可知，最重要的特征不是借款人的个人信用情况和资产情况，而是借款人的个人征信被查询次数(inq.last.6mths)，该指标反映的是借款人在最近六个月之内的贷款活动的频率。一个可能的解释是当借款人越是频繁地借款时，其违约的可能性越高。为了验证该猜测，我们画出不同查询次数的逾期比例，并进行线性拟合，结果如图 6-2 所示。

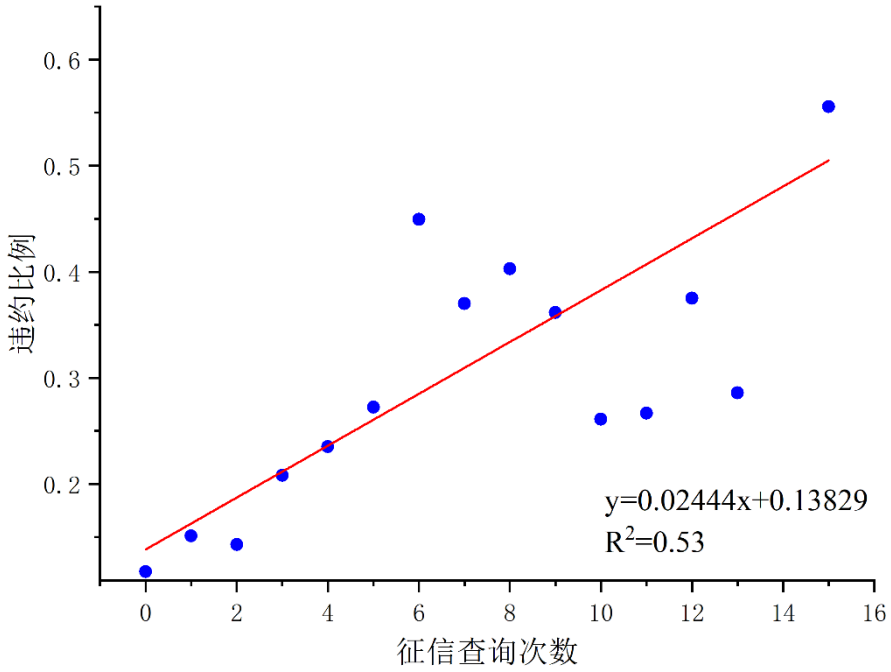

图 6-2 关于征信查询次数和违约比例线性拟合结果

线性拟合的结果显示，借款人在过去六个月内的个人征信被查询次数和违约比例呈正相关关系，说明上述猜测是正确的，即借款人一段时期以内借款活动越频繁借款人违约风险越大。因此，inq.last.6mths 是一个我们过去忽略但是很重要的指标。

6.2.2 是否达到借款标准

有趣的一点是 credit.policy(3.2%)指标，即借款人的担保标准是否满足公司要求，在该模型在的重要性并没有人们预期的那么大。为了探究是否满足公司要求对是否完全还款的影响，我们分析数据集中的真实情况，如图 6-3 所示。

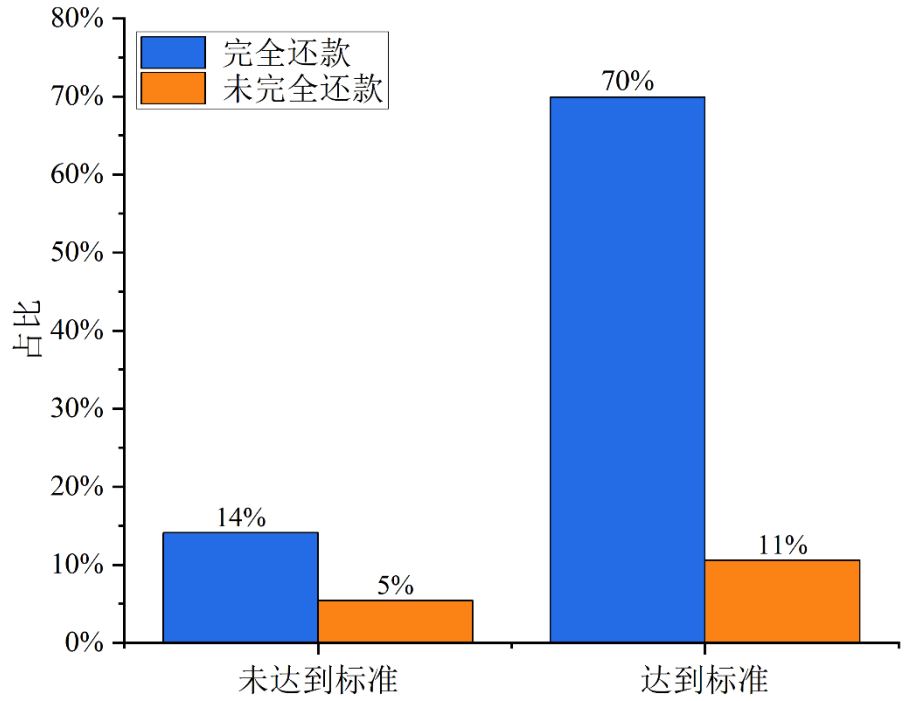

图 6-3 是否达到公司借款标准中的还款情况

通过图中数据我们可以看到，在未达到公司借款标准的群体中(19%)，有 14%的人会完全还款，这说明在无法达到现有的借款标准的人群中同样存在优质客户，公司采用的现有借款标准并不能有效对顾客进行有效分类，该群体在过去是被忽略的，这需要我们采取更合适的指标进行识别。

6.3 结论

本文采用 OBB 的方式在Loan Data 数据集上得到了特征重要性排序，根据对排序结果的分析，我们得出以下结论：

1) 在过去 6 个月内借款人的征信被查询次数是预测借款人是否会完全还款的一个重要指标，该指标的重要性在过去未被重视。
2) 在现有的借款标准下，未能满足该条件的群体中也同样存在优质客户，说明现有借款标准无法有效地对客户进行细分。

7 总结与展望

7.1 结论

欺诈检测是一项很困难的任务，因此一个很重要的领域是开发高效的欺诈检测方法。随着以深度学习为代表的机器学习方法在各个领域显示出巨大的应用潜力，采用机器学习方法进行欺诈检测成为新的研究思路。因此、本文的研究目的是探究一种新的机器学习方法深度Boosting 决策算法在欺诈检测中的应用。为了兼顾泛化性和可解释性，本文将神经网络与决策树进行融合构建弱分类器，并将之用于Boosting 算法。

本文在三个真实数据集上进行实验并和 4 种Boosting 算法作为比较，并引入多种不同的评价。结果证明了深度Boosting 决策树算法整体性能优于这 4 种算法，同时发现深度Boosting 决策算法的一个特点是精度高，但识别能力相对弱。同时本文又探究了一些重要的模型超参数对模型性能的影响，包括为调参工作总结了一些经验。

最后为了尝试对模型进行了解释，使用 OBB 方法在一个数据集上得到了特征重要性排序，并发现了一种过去被人们忽略的一种重要特征 inq.last.6mths（债权人在过去 6 个月内对借款人的查询次数），同时我们还发现在不满足借贷担保标准的人群中也存在部分优质客户，这部分群体在过去通常会被忽略，因此启发我们应改进现有的借款标准。

7.2 局限与未来研究方向

同时本文研究也存在一些不足，针对这些局限，可以在未来展开更多的研究。

1) 本文构造的算法选取了 Adam 优化算法，并没有深入探究其他优化算法对模型性能的影响，因此未来可以在该领域进行更多的探索。

2) 该模型针对非均衡分类问题的性能虽然整体优于现有算法，但我们认为仍然存在性能提升的空间，尤其是可以在本文算法的基础上借鉴处理非均衡分类问题的策略进行改进。

3) 虽然我们对模型的可解释性做了尝试，但仍没有完全打开模型，例如仍然无法针对诸如“为什么这个人欺诈的概率会比别人高？”给出一个合理解释，本文认为未来可以从模型偏见和统计错分样本的角度更加深入地研究模型的可解释性问题。

参考文献

[1] 中国信息通信研究院. 中国数字经济发展白皮书[R].2021.

[2] 李玉泉, 乔石. 大金融背景下保险欺诈的新特征和风险防范路径[J]. 保险研究, 2021(04): 121-127.

[3] BAO Y, KE B, LI B, et al. Detecting accounting fraud in publicly traded US firms using a machine learning approach[J]. Journal of Accounting Research, 2020,58(1): 199-235.

[4] 肖可砾, 熊辉. 运用数据挖掘技术检测金融欺诈行为[J]. 金融电子化, 2010(08): 89-90.

[5] KOTSIANTIS S, KANELLOPOULOS D, PINTELAS P. Handling imbalanced datasets: A review[J]. GESTS International Transactions on Computer Science and Engineering, 2006,30(1): 25-36.

[6] JO T, JAPKOWICZ N. Class imbalances versus small disjuncts[J]. ACM Sigkdd Explorations Newsletter, 2004,6(1): 40-49.

[7] MIERSWA I. Controlling overfitting with multi-objective support vector machines: Proceedings of the 9th annual conference on Genetic and evolutionary computation[C], 2007.

[8] CHAWLA N V, BOWYER K W, HALL L O, et al. SMOTE: synthetic minority over-sampling technique[J]. Journal of artificial intelligence research, 2002,16: 321-357.

[9] 周志华. 机器学习[M]. 2016年1月第1版. 北京: 清华大学出版社, 2016.

[10] ZHOU Z. Ensemble methods: foundations and algorithms[M]. CRC press, 2012.

[11] SEIFFERT C, KHOSHGOFTAAR T M, Van HULSE J, et al. RUSBoost: A hybrid approach to alleviating class imbalance[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2009,40(1): 185-197.

[12] CHAWLA N V, LAZAREVIC A, HALL L O, et al. SMOTEBoost: Improving prediction of the minority class in boosting: European conference on principles of data mining and knowledge discovery[C]: Springer, 2003.

[13] GUO H, VIKTOR H L. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach[J]. ACM Sigkdd Explorations Newsletter, 2004,6(1): 30-39.

[14] RAYHAN F, AHMED S, MAHBUB A, et al. Cusboost: Cluster-based under-sampling with boosting for imbalanced classification: 2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)[C]: IEEE, 2017.

[15] FAN W, STOLFO S J, ZHANG J, et al. AdaCost: misclassification cost-sensitive boosting: Icml[C]: Citeseer, 1999.

[16] TING K M. An empirical study of MetaCost using boosting algorithms: European Conference on Machine Learning[C]: Springer, 2000.

[17] YING LIU X, WU J, HUA ZHOU Z. Exploratory undersampling for class-imbalance learning[J]. ICDM 2006, 2006.

[18] FREUND Y, SCHAPIRE R E. A decision-theoretic generalization of on-line learning and an application to boosting[J]. Journal of computer and system sciences, 1997,55(1): 119-139.

[19] CHEN T, GUESTRIN C. Xgboost: A scalable tree boosting system: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining[C], 2016.

[20] FRIEDMAN J H. Greedy function approximation: a gradient boosting machine[J]. Annals of statistics, 2001: 1189-1232.

[21] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. Nature, 2015,521(7553): 436.

[22] 李航. 统计学习方法[M]. 第2版. 北京: 清华大学出版社, 2012.

[23] FROSST N, HINTON G. Distilling a Neural Network Into a Soft Decision Tree[J]. arXiv: 1711-9784,

2017.
[24] KONTSCHIEDER P, FITERAU M, CRIMINISI A, et al. Deep Neural Decision Forests: 2015 IEEE International Conference on Computer Vision (ICCV)[C], 2015.
[25] 李思瑶, 王积田, 柳立超. 基于生存分析的P2P网络借贷违约风险影响因素研究[J]. 经济体制改革, 2016(6): 156-160, 共5页.