中央财经大学

Central University of Finance and Economics

课　　程　统计计算

论文题目　基于集成算法模型的信贷信用评价

班　　级　17 统计专硕班

姓　　名　冯洋洋 甘鹏程 王思雨

指导教师　杨玥含

论文日期　2017/01/10

目录

摘要：随机森林、bagging、adaboost 等方法都是基于弱分类器的集成算法，本文针对信贷数据对这三种算法进行调试，并比较最终预测结果和这三种算法的预测效果和机器性能占用。本次试验中仅需要 8 次。而且运行时间上 XGBoost 的多线程并行运行，也发挥了极大作用，比 Adaboost 缩短时间达到了近 100 倍，而比随机森林缩短时间近 200 倍。三种集成分类器预测准确率均在 80%左右，但考虑到准确度的敏感性，XGBoost 表现最优。

关键词：随机森林 集成算法 AdaBoost XGBoost

一、绪论

1.1 研究背景

早在 1997 年, 国际机器学习界的权威 T.G. Dietterich 就将集成学习列为机器学习四大研究方向 之首。最近几年, 在机器学习、神经网络、统计学等领域的很多研究者都投入到集成学习的研究中, 使得该领域成为了一个相当活跃的研究热点。现在已经有很多集成学习算法, 比如: Bagging 算法、Boosting 算法、Arcing 算法、Random Forest 算法、GASEN 算法等等。为便于研究针对某一具体信贷数据的不同集成算法特点和效果,本文根据选择过程中核心策略的特征将选择性集成算法分为三类,即 Adaboost,XGBoost,随机森林;然后利用 UCI 数据库的 Credit 数据集,从预测性能的不同方面对这些典型算法进行了实验比较;最后总结了各类方法的优缺点, 并展望集成算法的未来研究重点。

1.2 研究意义与目的

随机森林、bagging、adaboost 等方法都是基于决策树的组合方法，但是这些方法和传统的统计思想有很大不同，如果针对具体数据集调整除合适模型来进行预测是本文的核心，特别是针对信贷数据需要把握的特点即精确度的要求，对这三种算法进行调试，并比较最终预测结果和这三种算法的优缺点。不论是 boosting 还是 bagging 当中，当使用的多个分类器的类型都是一致的，如何针对数据集特征挖掘不同算法和分类器的优势才能选择好的分类器。通过这篇文章，对这三种集成算法可以有更清晰的了解。

1.3 文献综述

由于银行信贷违约与否有着现实需求，使得大量学者对个人信用评价进行了研究。所使用的研究技术与手段从传统的统计学、运筹学方法逐步过度到了数据挖掘、机器学习领域的最新成果李航[1]《统计学习方法》中对机器学习算法有着详细地讲述。根据信用评价目的不同也分成诸多领域，其中最普遍的也是最直接的就是根

据信用可用户的人口学信息，以往信用使用记录以及用户经济能力来判别信用卡用户在未来是否会存在违约行为或者违约行为的概率是多大。

决策树是一种非参数统计方法，决策树方法具有自动选择变量、较好地处理缺失信息、准确性较高优点，李旭升[6](2006)对多种贝叶斯网络模型在个人信用领域的应用进行了研究，结果表明贝叶斯网络具有预测精度高、稳健性好的优点。

二十一世纪以后，计算机和信息技术得到了极大的发展，神经网络、遗传算法等人工智能方法成为个人信用评分研究的前沿。Sun 和 Li[7](2008）使用加权投票法对多重判别、logistic 回归，神经网络、决策树、支持向量机以及最近邻模型的预测结果进行了组合，他们认为组合模型预测总精度和稳健性都得到了提高这也是集成算法的雏形。

除了利用强学习机来建立信用评价模型，还有一种思路是通过 bagging 或 boosting 以及 boosting 提升算法等产生多个训练集，并选取某种不稳定的分类算法（即训练集的微小变动能够使得分类结果显著变动，如决策树、神经网络等）在这些训练集上建立模型，最后对这些模型预测结果进行个人违约与否的投票组合。这种组合建模方法也称为集成或融合，David West[9]（2005）采用和方法构建了神经网络模型集成模型，Finlay,S.H 使用和集成了决策树模型，他们认为集成方法可以显著提高信用评分模型的预测精度及泛化能力。Finlay 和 Steven[10]建立了多种 bagging 和 boosting 集成个人信用评分模型，并将它们的应用效果与传统单一模型进行了比较，结果表明集成模型要明显优于单一模型。

1.4 论文结构

本文首先对几种常用的分类算法进行了简单的介绍。

1. 介绍了决策树算法。决策树是随机森林、AdaBoost 和 XGBoost 算法的基础，这三种算法的基分类器基本上都是决策树。

2. 介绍了随机森林算法。随机森林指的是利用多棵树对样本进行训练并预测的一种分类器。该分类器最早由 Leo Breiman 和 Adele Cutler 提出。

3. 继续介绍了 AdaBoost 算法，对算法的思想和原理进行了详细的描述。

4. 最后详细介绍了 XGBoost 算法。

然后，针对同一份数据集，本文分别用上述三种集成算法进行了训练，并计算出了预测结果。

最后，通过计算三种集成算法的准确率、精确度、召回率、算法运行时间和算法耗用内存大小五个指标，对决策树和三种集成算法进行了比较。

二、研究方法与创新点

2.1 研究方法

本文选取了 UCI 数据集中的台湾信用贷款数据 credit card clients Data Set，以信贷评价为研究对象，想要通过以往的信用贷款的历史数据（包括个人信息数据、个人能力数据以及以往交易历史数据）来建立未来信用评价模型，来帮助银行来较为准确的判断信用卡用户未来的是否违约状况。

1. 进行了关于数据方面的预处理，使得数据可以在 R 环境下比较好的工作。并按 80%和 20%的比例随机的划分为模型训练集以及测试集。

2. 分别采用四中建模方法，决策树，随机森林，AdaBoost 提升法，XGBoost 提升法对台湾信贷数据进行回归建模。并比较模型的详细结果参数，运算效率，占用内存等。

3. 通过对比选择最适合信用评价的集成算法模型，最后根据数据特点进行改进与提升，使模型的实用性更强。

2.2 创新点

对于信贷评价的研究，国内大多是构建综合评价模型，这是一种无监督的评价模型，单纯基于数据的评级，无法反映历史数据信息，大部分是基于有关专家或是客户经理的个人经验对未来是否违约进行判断。精度较差，而且没有充分提取相关特征，缺乏对相关问题的针对性，容易受到数据收集不完全的影响。 本文的创新之处就在于利用近年来发展较为迅速的统计学习算法——集成算法来建立信用评价模型，使得银行可以有效利用信息来进行管理。分别利用了三种集成学习算法，包括，随机森林，Adaboost 以及 XGBoost 三种。其中，XGBoost 算法还进行了根据叶子节点增加新的特征变量来提高模型判断的准确率。并比较了三种集成方法，以及三种方法在评价信用贷款数据上的表现与效率。 创新点还在于根据数据的实际意义——信贷数据进行建模，因为是关系到银行利益，要增加对违约判别的准确率，所以在三个模型中都有进行调整阈值的过程，虽然降低了总体的模型准确率，但建立的模型更加具有现实意义，提高了真阳性率可以很好的帮助银行提前预知用户的违约情况，规避风险。

三、分类算法描述

本章分别介绍了随机森林、AdaBoost 和 XGBoost 三种集成算法，利用同一个数据集，从不同维度对三种算法的效率进行了比较。

3.1 决策树和随机森林

决策树是随机森林的基础，理解决策树的原理，对于理解随机森林算法具有很大帮助。

3.1.1 决策树

决策树又称为判定树，是运用于分类的一种树结构，其中的每个内部节点代表对某一属性的一次测试，每条边代表一个测试结果，叶节点代表某个类或类的分布。决策树的决策过程需要从决策树的根节点开始，待测数据与决策树中的特征节点进行比较，并按照比较结果选择选择下一比较分支，直到叶子节点作为最终的决策结果。

3.1.2 决策树的学习过程

- 特征选择：从训练数据的特征中选择一个特征作为当前节点的分裂标准（特征选择的标准不同产生了不同的特征决策树算法）- 决策树生成：根据所选特征评估标准，从上至下递归地生成子节点，直到数据集不可分则停止决策树停止生长。

- 剪枝：决策树容易过拟合，需要剪枝来缩小树的结构和规模（包括预剪枝和后剪枝）。

3.1.3 随机森林

3.1.3.1 随机森林简介

作为新兴起的、高度灵活的一种机器学习算法，随机森林（Random Forest，简称 RF）拥有广泛的应用前景，其实从直观角度来解释，每棵决策树都是一个分类器（假设现在针对的是分类问题），那么对于一个输入样本，N 棵树会有 N 个分类结果。而随机森林集成了所有的分类投票结果，将投票次数最多的类别指定为最终的输出，这就是一种最简单的 Bagging 思想。

3.1.3.2 随机森林特点

随机森林是一种很灵活实用的方法，它有如下几个特点[5]：

- 在当前所有算法中，具有极好的准确率
- 能够有效地运行在大数据集上
- 能够处理具有高维特征的输入样本，而且不需要降维
- 能够评估各个特征在分类问题上的重要性
- 在生成过程中，能够获取到内部生成误差的一种无偏估计

- 对于缺省值问题也能够获得很好得结果

3.1.3.3 随机森林的相关算法知识

随机森林看起来是很好理解，但是要完全搞明白它的工作原理，需要很多机器学习方面相关的基础知识。在本文中简单谈一下，而不逐一进行赘述。

- 信息、熵以及信息增益的概念

这三个基本概念是决策树的根本，是决策树利用特征来分类时，确定特征选取顺序的依据。

对于机器学习中的决策树而言，如果带分类的事物集合可以划分为多个类别当中，则某个类x_i的信息可以定义如下:

$$I(X = x_i) = -log_2 p(x_i)$$

x_i用来表示随机变量的信息，$p(x_i)$指是当x_i发生时的概率。

熵是用来度量不确定性的，当熵越大，X=xi 的不确定性越大，反之越小。对于机器学习中的分类问题而言，熵越大即这个类别的不确定性更大，反之越小。

信息增益在决策树算法中是用来选择特征的指标，信息增益越大，则这个特征的选择性越好。

- 集成学习

集成学习通过建立几个模型组合的来解决单一预测问题。它的工作原理是生成多个分类器/模型，各自独立地学习和作出预测。这些预测最后结合成单预测，因此优于任何一个单分类的做出预测。

随机森林是集成学习的一个子类，它依靠于决策树的投票选择来决定最后的分类结果。

3.1.3.4 随机森林的生成

前面提到，随机森林中有许多的分类树。森林中的每棵树都是独立的，99.9%不相关的树做出的预测结果涵盖所有的情况，这些预测结果将会彼此抵消。少数优秀的树的预测结果将会超脱于芸芸“噪音”，做出一个好的预测。将若干个弱分类器的分类结果进行投票选择，从而组成一个强分类器，这就是随机森林 bagging 的思想（关于 bagging 的一个有必要提及的问题：bagging 的代价是不用单棵决策树来做预测，具体哪个变量起到重要作用变得未知，所以 bagging 改进了预测准确率但损失了解释性。）[4]。下图可以形象地描述这个情况：

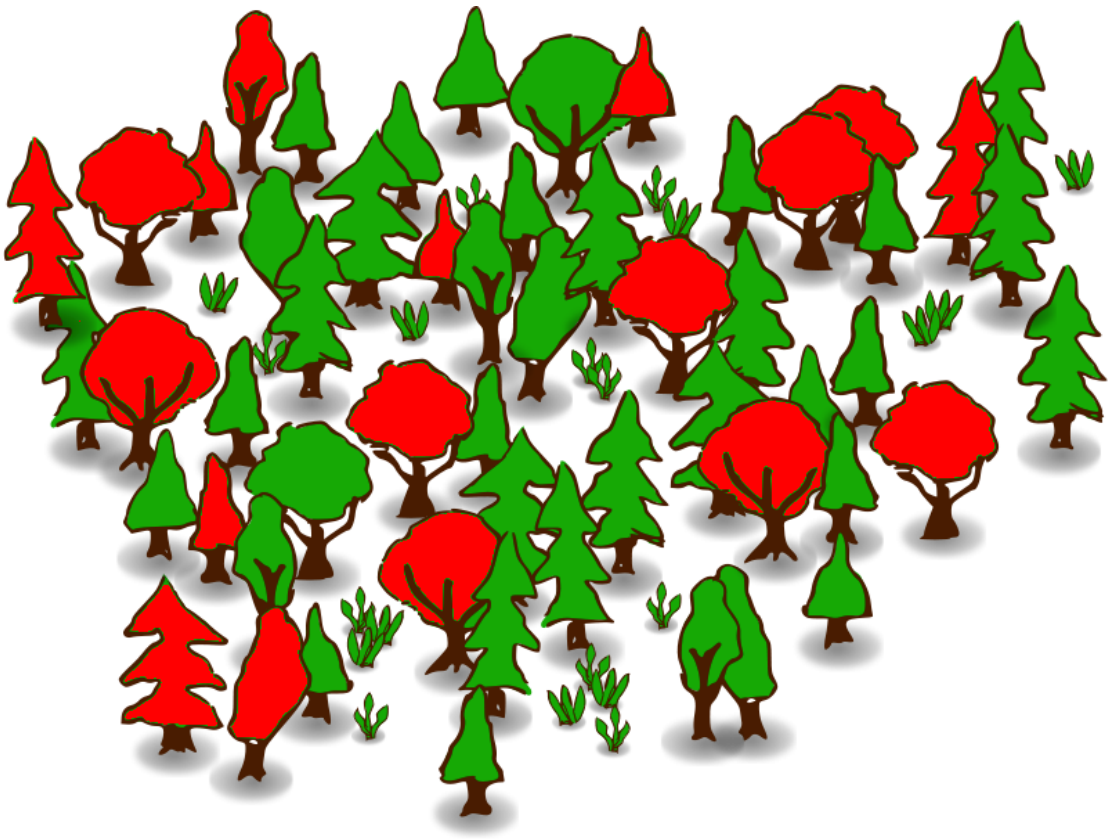

图1 随机森林示意图

- 如果训练集大小为N，对于每棵树而言，随机且有放回地从训练集中的抽取N个训练样本（这种采样方式称为bootstrap sample方法），作为该树的训练集

- 如果每个样本的特征维度为M，指定一个常数m<<M，随机地从M个特征中选取m个特征子集，每次树进行分裂时，从这m个特征中选择最优的；

- 每棵树都尽最大程度的生长，并且没有剪枝过程。

随机森林的分类效果（错误率）与两个因素有关

- 森林中任意两棵树的相关性：相关性越大，错误率越大；

- 森林中每棵树的分类能力：每棵树的分类能力越强，整个森林的错误率越低。

减小特征选择个数m，树的相关性和分类能力也会相应的降低；增大m，两者也会随之增大。所以关键问题是如何选择最优的m（或者是范围），这也是随机森林唯一的一个参数。

3.2 AdaBoost

提升（boosting）方法是一种常用的机器学习方法，应用广泛且有效。在分类问题中，它通过改变训练样本的权重，学习多个分类器，并将这些分类器进行线性组合，提高分类的性能。

3.2.1 提升方法的基本思路

提升方法基于这样一种思想：对于一个复杂的任务来说，将多个专家的判断进行适当的综合所得出的判断，要比其中任何一个专家单独的判断好。实际上，就是“三个臭皮匠赛过诸葛亮”的道理。

历史上，Kearns 和 Valiant 首先提出了“强可学习（strong learnable）”和“弱可学习（weakly learnable）”的概念[3]。指出：在概率近似正确学习的框架中，一个概念（一个类），如果存在一个多项式的学习算法能够学习它，并且正确率很高，那么就称这个概念是强可学习的；一个概念，如果存在一个多项式的学习算法能够学习它，学习的正确率仅比随机猜测略好，那么就称这个概念是弱可学习的。非常有趣的是 Schapire 后来证明强可学习与弱可学习是等价的，也就是说，在 PAC 学习的框架下，一个概念是强可学习的充分必要条件是这个概念是弱可学习的。

这样一来，问题便成为，在学习中，如果已经发现了“弱学习算法”，那么能否将它提升（boost）为“强学习算法”。大家知道，发现弱学习算法比发现强学习算法容易得多。那么如何具体实施提升，便成为开发提升方法时所要解决的问题。关于提升方法的研究很多，有很多算法被提出。最具有代表性的是 AdaBoost 算法。

对于分类问题而言，给定一个训练样本集，求比较粗糙的分类规则（弱分类器）要比求精确的分类规则（强分类器）容易的多。提升方法就是从弱学习算法出发，反复学习，得到一系列弱分类器，然后组合这些弱分类器，构成一个强分类器。大多数的提升方法都是改变训练数据的概率分布（训练数据的权值分布），针对不同的训练数据分布调用弱学习算法学习一系列弱分类器。

这样，对提升方法来说，有两个问题需要回答：一是在每一轮如何改变数据的权值或概率分布；二是如何将弱分类器组合为一个强分类器。关于第 1 个问题，AdaBoost 的做法是，提高那些被前一轮若分类器分错的样本权值，而降低那些被正确分类样本的权值。这样一来，那些没有得到正确分类的数据，由于其权值的加大而受到后一轮的弱分类器的更大关注。于是，分类问题被一系列弱分类器“分而治之”。至于第 2 个问题，即弱分类器的组合，AdaBoost 采取加权多数表决的方法。具体地，加大分类误差率小的弱分类器的权值，使其在表决中起较大的作用，减小分类误差率大的弱分类器的权值，使其在表决中起较小的作用。

AdaBoost 的巧妙之处在于它将这些想法自然且有效地实现在一种算法里。

3.2.2 AdaBoost 算法

现在叙述 AdaBoost 算法。假设给定一个二分类的训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$$

其中，每个样本点由实例与标记组成。实例$x_i \in R^n$,标记$y_i \in \{-1,1\}$。AdaBoost 利用以下算法，从训练数据中学习一系列弱分类器或基本分类器，并将这些弱分类器线性组合为一个强分类器。

输入：训练数据集$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$,其中$x_i \in R^n$,标记$y_i \in \{-1,1\}$；弱分类算法。

输出：最终分类器$G(x)$。

（1）初始化训练数据的权值分布，

$$D_1 = (w_{11}, \cdots, w_{1i}, \cdots, w_{1N}), w_{1i} = \frac{1}{N}, i = 1,2, \cdots, N$$

（2）对$m = 1,2, \cdots, M$ （a）使用具有权值分布D_m的训练数据集学习，得到基本分类器

$$G_m(x): R^n \to \{-1,1\}$$

（b）计算$G_m(x)$在训练数据上的分类误差率

$$e_m = \sum_{i=1}^{N} P\left(G_m(x) \neq y_i\right) = \sum_{i=1}^{N} w_{mi}\, I(G_m(x) \neq y_i)$$

(1)

（c）计算$G_m(x)$的系数

$$\alpha_m = \frac{1}{2} ln \frac{1 - e_m}{e_m}$$

(2)

（d）更新训练数据集的权值分布

$$D_{m+1} = (w_{m+1,1}, \cdots, w_{m+1,i}, \cdots, w_{m+1,N})$$

(3)

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} e^{-\alpha_m y_i G_m(x_i)}, i = 1,2, \cdots, N$$

(4)

这里，Z_m是规范化因子

$$Z_m = \sum_{i=1}^{N} e^{-\alpha_m y_i G_m(x_i)}$$

(5)

它使D_{m+1}成为一个概率分布。

（3）构建基本分类器的线性组合

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

(6)

得到最终的分类器

$$G(X) = sign(f(x)) = sign(\sum_{m=1}^{M} \alpha_m G_m(x))$$

(7)

对 AdaBoost 算法作如下说明：

步骤（1） 假设数据集具有均匀的权值分布，即每个训练样本在基本分类器中作用相同，这一假设保证第 1 步能够在原始数据集上学习基本分类器$G_1(x)$。

步骤（2） AdaBoost 反复学习基本分类器，在每一轮$m = 1,2,\cdots,M$中顺次地执行下列操作：

（a）使用当前分布D_m加权的训练数据集，学习基本分类器$G_m(x)$。

（b）计算基本分类器$G_m(x)$在加权训练数据集上的分类误差率：

$$e_m = \sum_{i=1}^{N} P\left(G_m(x_i) \neq y_i\right) = \sum_{G_m(x_i) \neq y_i} w_{mi}$$

(8)

这里，w_{mi}表示第m轮中第i个实例的权值，$\sum_{i=1}^{N} w_{mi} = 1$。这表明，$G_m(x)$在加权的训练数据上的分类误差率是被$G_m(x)$误分类样本的权值之和，由此可以看出数据权值分布$D_m$与基本分类器$G_m(x)$的分类误差率的关系。

（c）计算基本分类器$G_m(x)$的系数α_m，α_m表示$G_m(x)$在最终分类器中的重要性。由(2)可知，当$e_m \leq \frac{1}{2}$时，$\alpha_m \geq 0$，并且α_m随着e_m的减小而增大，所以分类误差率越小的基本分类器在最终分类器中的所用越大。

（d）更新训练数据的权值分布为下一轮作准备，式(4)可以写成：

$$w_{m+1,i} = \begin{cases} \frac{w_{mi}}{Z_m} e^{-\alpha_m} & = \quad G_m(x_i) = y_i \\ \frac{w_{mi}}{Z_m} e^{\alpha_m} & = \quad G_m(x_i) \neq y_i \end{cases}$$

由此可知，被基本分类器$G_m(x)$误分类样本的权值得以扩大，而被正确分类样本的权值却得以缩小。两相比较，由式(2)知误分类样本的权值被放大$e^{2\alpha_m} = \frac{1-e_m}{e_m}$倍。因此，误分类样本在下一轮学习中起更大的作用。不改变所给的训练数据，而不断改变训练数据的权值分布，使得训练数据在基本分类器的学习中起到不同的作用，这是 AdaBoost 的一个特点。

步骤（3） 线性组合$f(x)$实现M个基本分类器的加权表决。系数α_m表示了基本分类器$G_m(x)$的重要性，这里，所有α_m之和并不为 1.$f(x)$的符号决定实例x的分类，$f(x)$的绝对值表示分类的确信度。利用基本分类器的线性组合构建最终分类器是 AdaBoost 的另一个特点。

3.3 XGBoost

3.3.1 XGBoost 算法概要

XGBoost 是“极端梯度上升”(Extreme Gradient Boosting)的简称,从技术上说，XGBoost 是 Extreme Gradient Boosting 的缩写。它的流行源于在著名的 Kaggle 数据科学竞赛上被称为“奥托分类”的挑战。它可以处理多种目标函数，包括回归，分类和排序，是一个较为全面的分类器。 由于其他许多分类器，不管是强分类器或是集成分类器，在预测性能上的强大但是相对缓慢的实现，如上一章的 Adaboost 集成算法，不管是在在运行时间上还是在内存占有上开销都很大。XGBoost 成为很多比赛的理想选择。XGBoost 包还添加了做交叉验证和发现关键变量的额外功能。在优化模型时，这个算法使用了好几个参数。所以为了提高模型的表现，参数的调整十分必要。本节将讨论这些因素。

3.3.2 XGBoost 优势

XGBoost 算法总结起来大致其有三个优点：高效、准确度、模型的交互性[2]。

- 正则化：标准 GBDT 提升树算法的实现没有像 XGBoost 这样的正则化步骤。正则化用于控制模型的复杂度，对减少过拟合也是有帮助的。XGBoost 也正是以“正则化提升”技术而闻名。

- 并行处理：XGBoost 可以实现并行处理，相比 GBM 有了速度的飞跃。不过，需要注意 XGBoost 的并行不是 tree 粒度的并行，XGBoost 也是一次迭代完才能进行下一次迭代的（第 t 次迭代的代价函数里包含了前面 t-1 次迭代的预测值）。XGBoost 的并行是在特征粒度上的。决策树的学习最耗时的一个步骤就是对特征的值进行排序（因为要确定最佳分割点）。因此 XGBoost 在 R 重定义了一个自己数据矩阵类 DMatrix。XGBoost 在训练之前，预先对数据进行了排序，然后保存为 block 结构，后面的迭代中重复利用索引地使用这个结构，获得每个节点的梯度，大大减小计算量。这个 block 结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行。

- 高度灵活性：XGBoost 允许用户定义自定义优化目标和评价标准，它对模型增加了一个全新的维度，所以我们的处理不会受到任何限制。

- 缺失值处理：XGBoost 内置处理缺失值的规则。 用户需要提供一个和其它样本不同的值，然后把它作为一个参数传进去，以此来作为缺失值的取值。XGBoost 在不同节点遇到缺失值时采用不同的处理方法，并且会学习未来遇到缺失值时的处理方法。

- 剪枝：当分裂时遇到一个负损失时，传统 GBDT 会停止分裂。因此传统 GBDT 实际上是一个贪心算法。XGBoost 会一直分裂到指定的最大深度(max_depth)，然后回过头来剪枝。如果某个节点之后不再有正值，它会去除这个分裂。这种做法的优点，当一个负损失（如-2）后面有个正损失（如+10）的时候，就显现出来了。GBM 会在-2 处停下来，因为它遇到了一个负值。但是 XGBoost 会继续分裂，然后发现这两个分裂综合起来会得到+8，因此会保留这两个分裂。

- 内置交叉验证：XGBoost 允许在每一轮 boosting 迭代中使用交叉验证。因此，可以方便地获得最优 boosting 迭代次数。而传统的 GBDT 使用网格搜索，只能检测有限个值。

3.3.3 XGBoost 算法推导

XGBoost 在函数空间中用牛顿法进行优化。首先，boosting 是一种加法模型。XGBoost 同样属于 GBDT 梯度提升法，模型的基分类器都包含有树，对于给定的数据集 D={(x_i,y_i)}，XGBoost 进行 additive learning，学习 K 棵树，采用以下函数对样本进行预测。

$$\hat{y} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i) \qquad f_k \in F$$

这里 F 是函数空间，$f(x)$是回归树 CART。

$$F = \{f(x) = w_{q(x)}\}(q: R^m \to T, w \in R)$$

$q(x)$标识将样本 x 分到了某个叶子节点上，w是叶子节点的分数，（leaf score），所以$w_q(x)$ 表示回归树对样本的预测值。 回归树的预测输出是实数分数，可以用于回归，分类，排序等任务中，对于回归问题，可以直接作为目标值，对于分类问题，需要映射成概率，比如采用逻辑函数，然后可以控制阈值，进行两种分类错误的把控。 XGBoost 对传统的提升树算法 Adaboost 等的改进，在于在参数空间的目标函数中加入了正则化项，来惩罚模型的复杂程度，进而控制过拟合。和 Adaboost 一样都是通过最小化损失函数求解最优模型，并加入了阈值，如下公式所示：

$$L(\phi) = \sum_i l\left(\hat{y}_l, y_i\right) + \sum_k \Omega\left(f_k\right)$$

误差函数可以是 square loss，logloss 等，也可以自己定义损失函数，只要能够求出目标函数的梯度和 Hessian 矩阵，用户就可以自定义训练模型时的目标函数这也正是 XGBoost 的优势之一，可以通过研究目的的不同自己定义损失函数， 在公式（3-3）中，相比于原始的 GBDT，XGBoost 的目标函数多了正则项，是学习出来的模型更加不容易过拟合。衡量树的复杂程度主要与树的深度，内部节点的个数，叶子节点的个数，叶子节点的权重有关，因此 XGBoost 对这些参量进行了约束。得出了正则项为：

$$\Omega(f) = \gamma T + \frac{1}{2}\rho \parallel w \parallel^2$$

正则项对每棵树的复杂程度都应进行惩罚，对每个节点进行了复杂度的惩罚，从另一种角度来说也就进行了自动的剪枝。 另外，还可以选择使用线性模型替代树模型，从而得到带$L1 + L2$惩罚的线性回归正则项可以是$L1$正则，$L2$正则 。 第t次迭代后，模型的预测等于前$t-1$次的模型预测加上第 t 棵树的预测。将目标函数在前$t-1$次的模型y_i^{t-1}处进行泰勒展开，并将常数项去掉即得到：

$$L^{(t)} = \sum_{i=1}^{n}[\, g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

公式中，$g_i = \delta_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$　　$h_i = \delta^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ 把$f_t, \Omega(f_t)$写成树结构的形式，得到：

$$L^{(t)} = \sum_{i=1}^{n}[\, g_i w_{q(x_i)} + \frac{1}{2} h_i w^2_{q(x_i)}] + \gamma T + \rho \frac{1}{2} \sum_{j=i}^{T} w_j^2$$

则目标函数可以写成按叶节点累加的形式：

$$L^{(t)} = \sum_{j=1}^{T}[\, G_i w_j + \frac{1}{2}(H_j + \rho) w_j^2] + \gamma T$$

如果确定了树的结构（即$q(x)$确定），为了使目标函数最小，可以令其导数为0，解得每个叶节点的最优预测分数为：

$$W_j^* = -\frac{G_j}{H_j + \rho}$$

$$\hat{L}^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \rho} + \gamma T$$

公式（3-9）的负部衡量了每个叶子节点对总体损失的的贡献，我们希望损失越小越好，则公式的（3-9）的负部值越大越好。 因此，对一个叶子节点进行分裂，分裂前后的增益定义为：

$$Gain = \frac{G_L^2}{H_L + \rho} + \frac{G_R^2}{H_R + \rho} - \frac{(G_L + G_R)^2}{H_L + H_R + \rho} - \gamma$$

Gain 的值越大，分裂后 L 减小越多。所以当对一个叶节点分割时，计算所有候选特征值所对应的 gain，选取 gain 最大的进行分割。但由于精确遍历所有可能的分割点是效率很低的，所以，实际上 XGBoost 采用的是对于每个特征，不是简单地按照样本个数进行分位，而是以二阶导数值作为权重，进行分位点的选择，以此减少计算复杂度。在学习每棵树前，提出候选切分点，这也是 XGBoost 可以实现并行的原因之一，可以提前切割分位点。 最后，XGBoost 算法还借鉴了 bagging 的 bootstrap 自助法行抽样，还借鉴了随机森林的列抽样，即特征抽样。这样减少过拟合同时还降低了计算复杂度。

四、实证分析

本章利用信贷数据建立了三种集成算法的模型，计算了模型的准确度、精确度、召回率、算法耗用的时间和内存。针对每一种算法，本文的绘制了其对应的 ROC 曲线，ROC 曲线可以帮助我们理解模型的效率。

4.1 数据介绍与处理

4.1.1 数据介绍

文本使用的数据是从 UCI 数据集上下载的 default of credit card clientsu 数据集。数据一共有 30000 行记录，24 个属性。数据不包含缺失值，因此简化了预处理过程。数据集的各个变量介绍如下表。

表 1 变量介绍

变量名	变量说明	变量类型	取值范围	注
X1	给定额度	数值型	1 万-100 万	
X2	性别	分类型	1:男，2:女	
X3	教育水平	分类型	1:graduate school;2:university;3:high school;4:others	
X4	婚姻状况	分类型	1:married;2:single;3:others	
X5	年龄	数值型	21-79	
X6-X11	还款状态	数值型	<0:按时还款，>0:超期相应月数还款	
X12-X17	票据金额	数值型	X12 表示 2015-09，以此类推	
X18-X23	先前付款额	数值型	X18 表示 2015-09，以此类推	
Y	**是否违约**	**分类型**	**0:未违约，1:违约**	

4.1.2 数据预处理

本数据比较干净，没有缺失值，因此对数据的处理重点放在数据转换上。前面已经介绍，性别、教育水平、婚姻状况、还款状态和是否违约都是分类型数据，因此在 R 中建模的时候要将这些变量转换为因子型。

由于因变量中违约（1）和未违约（0）的比例相差很大，因此本文采用分层抽样的方法，从原始数据中抽取 20%的数据作为测试集，剩下的 80%的数据作为训练集。

由于 XGBoost 包在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂。而这又恰恰是各种 GBDT 算法中最耗时间的。所以 XGBoost 想到将各个特征的增益计算就进行多线程进行。那么在定义数据结构时，XGBoost 在 R 重定义了一个自己数据矩阵类 DMatrix。因此，预处理过程和随机森林和 Adaboost 算法不同，XGBoost 在训练之前，首先要进行数据结构的处理。

- 首先按将数据按 80%和 20%的概率随机抽样划分训练集 train 与测试集 test。

- 由于 XGBoost 要求所有的特征变量都应该为数值变量，所以首先将分类变量 sex，education 以及 marriage 进一步处理为独热编码 one hot。并将 one hot 数值变量代替原始分类变量。

- 将编码后的训练样本矩阵以及测试样本矩阵都进行整体化的数值处理，将学习标签“是否违约”也处理成 0-1 的数值变量。

经过 XGBoost 数据处理的用于构建模型的数据结构一共产生 4 个，包括训练数据集 train，训练标签 train_label，测试数据集 test，测试标签 test_label。都是 DMatrix 类型数据。

4.2 描述分析

本部分主要关注不同人口学属性与违约状况、授信额度的关系。

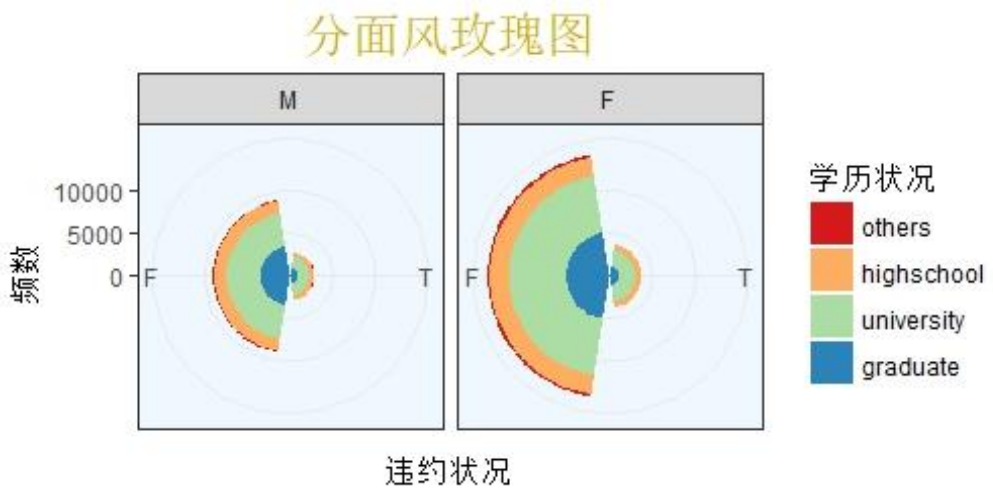

图 2 违约状况分面风玫瑰图

从图中可以看出，不管是男性还是女性，信贷用户中大学毕业生人数比较多，其次是研究生。其他学历人数较少，还可以看出男性的信贷违约人数要少于女性，说明女性更容易违约。在途中还可以看出随着学历的逐渐上升，违约状况也有所改观，说明教育水平会提升信用状况。

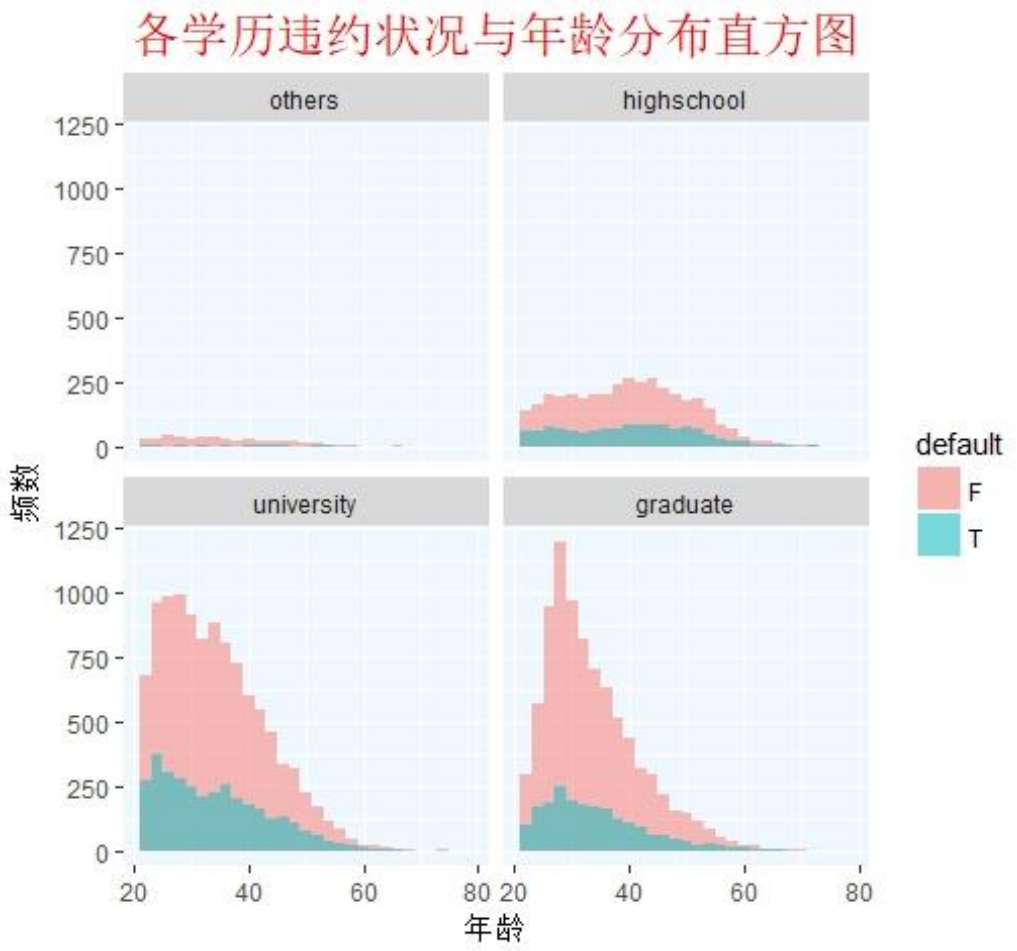

图3 各学历违约状况与年龄分布直方图

根据所受教育水平的不同，违约状况的的年龄分布也有所不同，在学历为其他的用户中违约的年龄分布比较均匀，在高中毕业的水平下，总体的年龄分布比较均匀，违约人的年龄分布也是比较均匀的，在大学的教育水平下，年龄分布比较不均匀，整体成右偏分布，而违约人的年龄分布峰较低，比较平缓。在研究生阶段的的年龄分布峰较高，成尖峰分布，且有右偏拖尾。整体的违约比率较低，分布比较平缓。

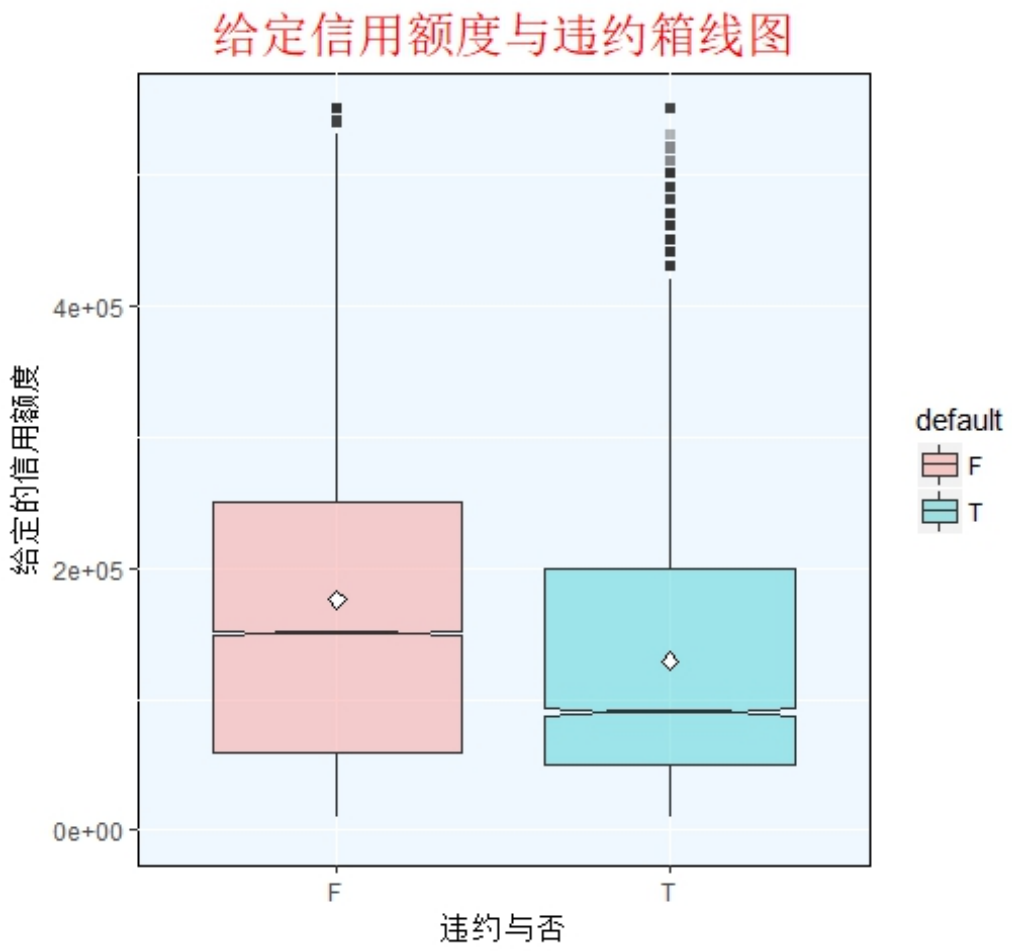

图 4 给定信用额度与违约箱线图

如图所示，白色的点代表着中位数，而象限图中的白线则代表着均值。对于给定的信用额度与违约的关系上来看，可以看出的关系是，对于未来不违约的人通常银行给予的信用额度是比较高的，说明银行在当时信用分配筛选时，是下过一番功夫的。因此会导致未来违约的人的信用额度偏低。

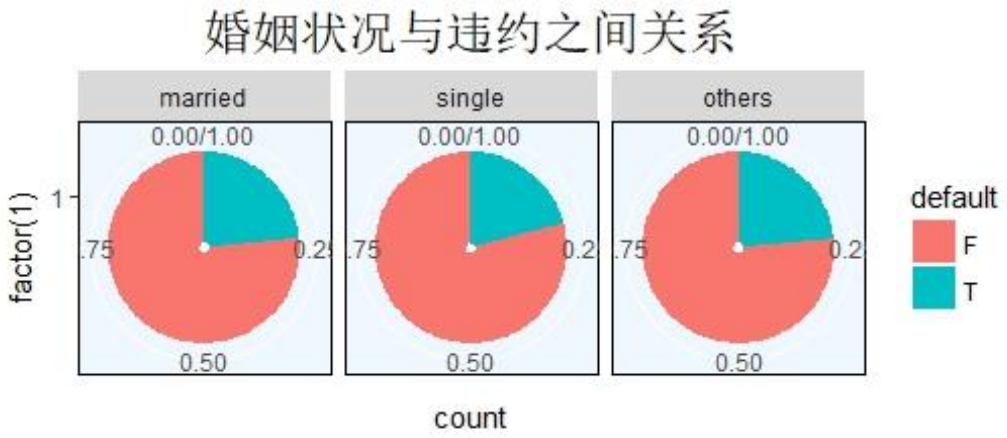

图 5 婚姻状况与违约的关系

在图中可以看出婚姻状况与违约之间的关系，数量关系没有描述统计的必要，主要进行百分比之间的转化，在分面饼状图中可以清楚的看出，婚姻状况对违约并没有很大的影响，主要的影响是比较细微的，单身的违约概率是比较小的，大约占到总体的 1/5 左右，而已婚和其他的状况违约概率是几乎一样的，都占各自总体的 1/4 左右。

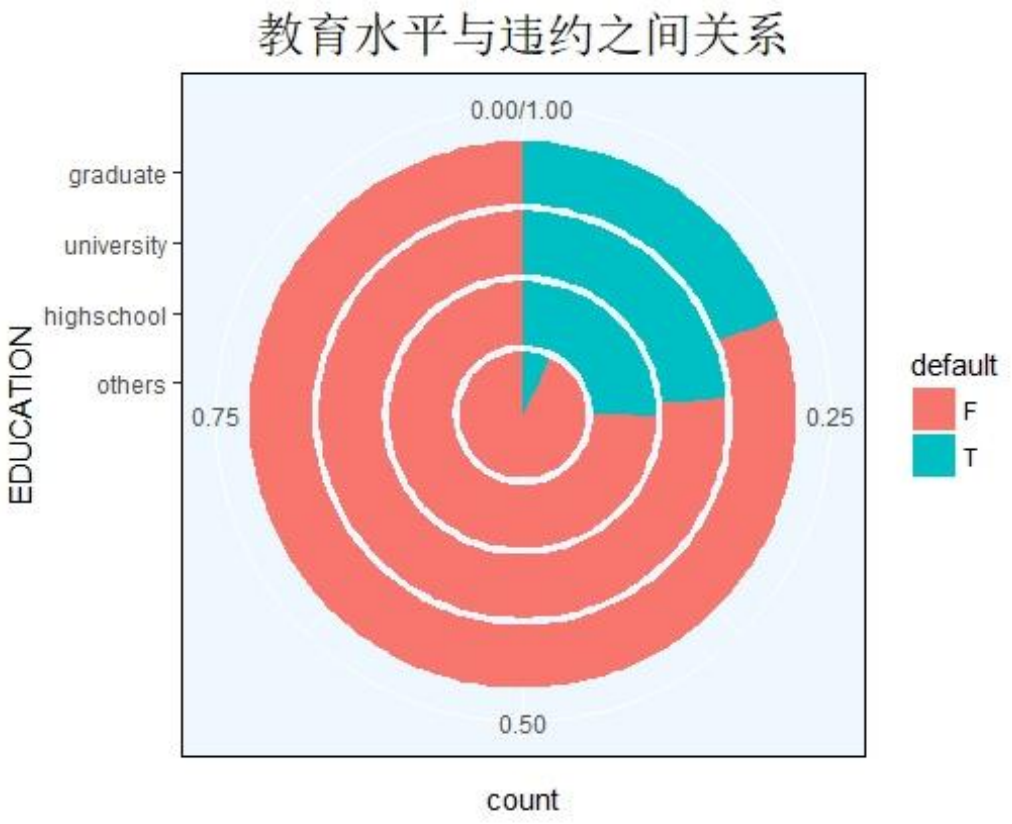

图 6 教育水平与违约的关系

在教育水平与违约状况环形图中，也可以看到，违约占比较高的教育水平为高中阶段，其次是大学，然后是研究生，相比之下，其他的教育水平违约占比较低，因此我们也可以大胆的推断 other 是一种更高的教育水平。有可能是博士阶段。各个学历水平的违约率都没超过 1/4，other 更是小于 1/10。

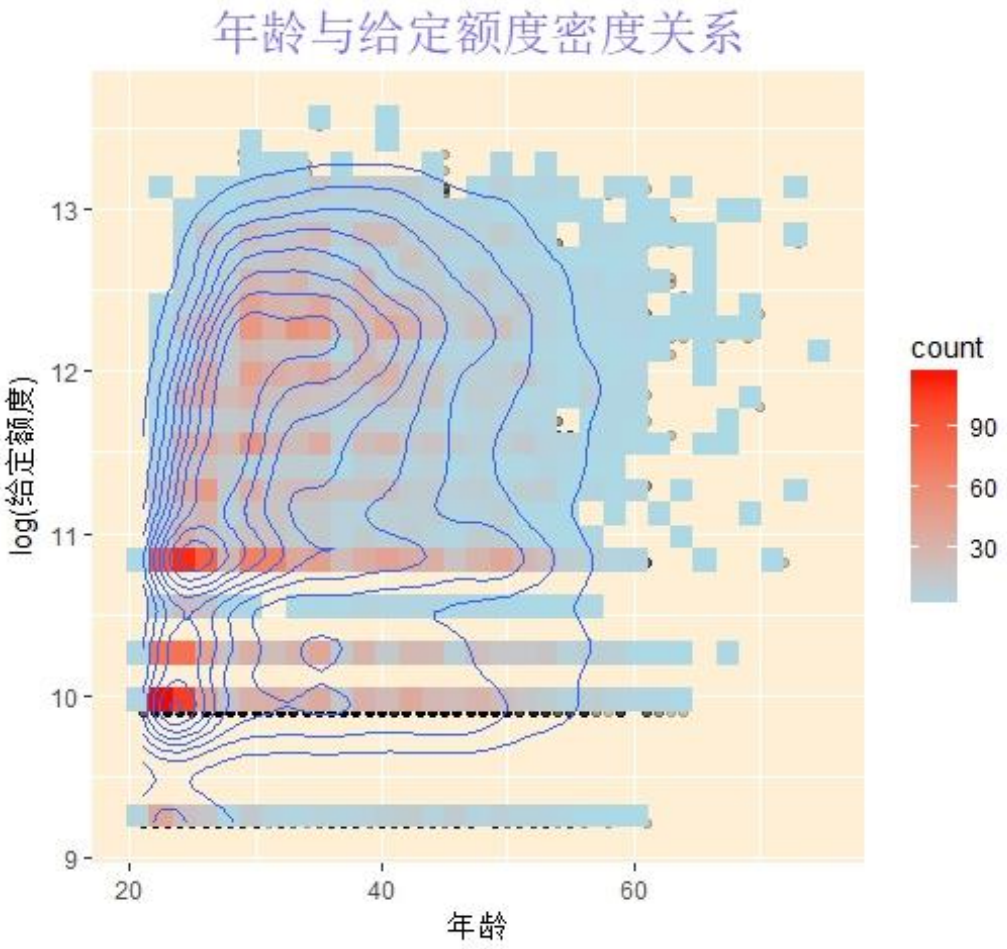

图 7 年龄与给点额度的密度关系

我们筛选的是具有违约状况的信用记录来进行分析。由于点的数量太多，因此绘制了分箱密度估计图，横坐标是违约用户的年龄，纵坐标是给定额度的对数值。来分析对于违约的用户，什么样的年龄和给定额度下违约的人数较多，可以在图中看到，红色区域对应着 25 岁给定额度为 10 的人数违约较多，其次还有 25 岁给定额度为 11 左右的人。因此银行在分配额度是应该注意这一点，利用密度图来规避风险，尽量少分配给 25 年龄段更多的信用。还可以看出给定额度与年龄的关系并不大，年龄小一样可以获得较高额度。

4.3 集成分类器实证检验

4.3.1 随机森林模型实证检验

4.3.1.1 模型介绍

针对已经划分好的数据集，利用 R 软件中的 Randomfrest 包进行建模，该包中 randomForest()函数用于构建随机森林模型，主要涉及 2 个重要参数：ntree 和

mtry。 前者指定随机森林所包含的决策树数目，默认为 500.通过建立模型后的误差图进行调整，通过误差图的变化我们可以清楚的看到误差率在树的数量达到 100 以后趋于稳定，因此数目定在 500 是合理的。

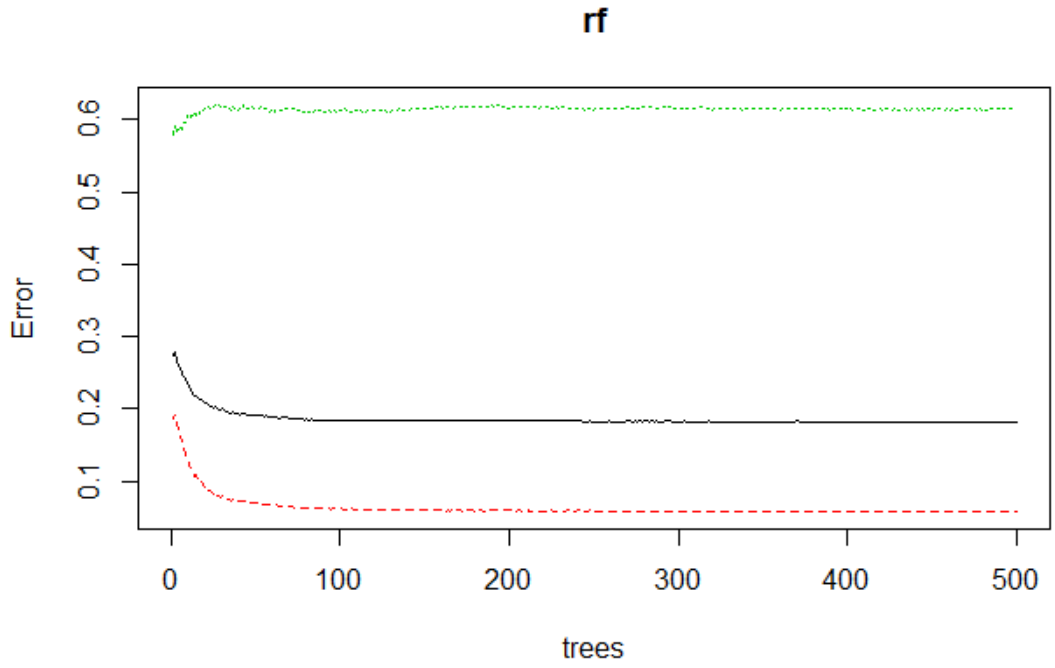

图 8 误差图

后者指定节点中用于二叉树变量的个数，默认情况一下一般取数据集变量个数的二次方根或者三分之一。在本文模型的调试中，依次在循环函数中计算所有可行值进行建模后的平均错误率，最后取最低值对应的 mtry 值。 实际结果中 mtry 的取值范围是【1，23】，平均错误率最低值为 0.2872026 对应 mtry 为 9. 所有变量都入选进了模型，其重要性的排名如下

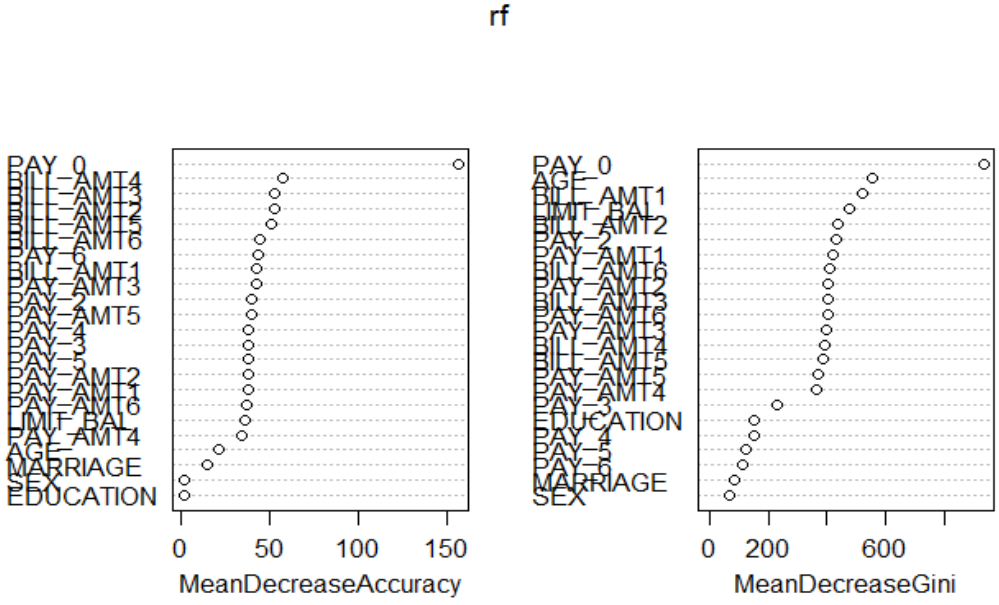

图 9 变量重要性

4.3.1.2 训练模型结果

在 ntree=500,mtry=9 的情况下进行随机森林建模，得到的混淆矩阵如下：

表 2 混淆矩阵

	0	1	class.error
0	17575	1079	**0.05784282**
1	**3283**	**2063**	**0.61410400**

在 24000 个训练集样本中，有 19638 个样本被模型正确分类；平均错误率为 0.2881929 值得注意的是在 1 分类中即违约分类中，错误率到达了 0.61410400，远高于在 0 分类中的错误率 0.05784282 可见训练集得出的预测结果并不理想

4.3.1.3 测试集验证结果

对 6000 个测试集样本的的预测结果中，有 4895 个样本记录被预测正确，预测的混淆矩阵如下：

表 3 混淆矩阵

pred1	**0**	**1**
0	4414	**809**
1	**296**	**481**

本文采用预测准确率，召回率，精确率度和 F 值对预测结果进行评价，评价结果如下：

表 4 评价结果

Accuracy	recall	precision	F_measure
0.8156667	0.372093	0.6185567	0.464666

- 预测准确率的定义是：对于给定的测试数据集，分类器正确分类的样本数与总样本数之比。也就是说有大约 81.57 的测试集样本被预测正确

- 精确率的定义是：“正确被检索的 item(TP)“占所有”实际被检索到的(TP+FP)″的比例。在本数据集中也就是实际违约样本数占被预测为违约的样本的占比，被预测为违约的样本大约有 61.86%实际上确实违约了。

- 召回率的定义是：“正确被检索的 item(TP)”占所有“应该检索到的 item(TP+FN)”的比例。在本数据集中也就是所有违约记录中被模型指出的比率。也就是说，所有实际违约的样本记录中，模型只发现了约 37.21%

- 一般来说前两个指标是矛盾的，即一个表现好另一个表现就会差一些，F 值是对召回率和精确度的甲醛调和平均，综合了两个指标的结果。在本模型中 F 值约为 0.46

- ROC 曲线和 AUC 值

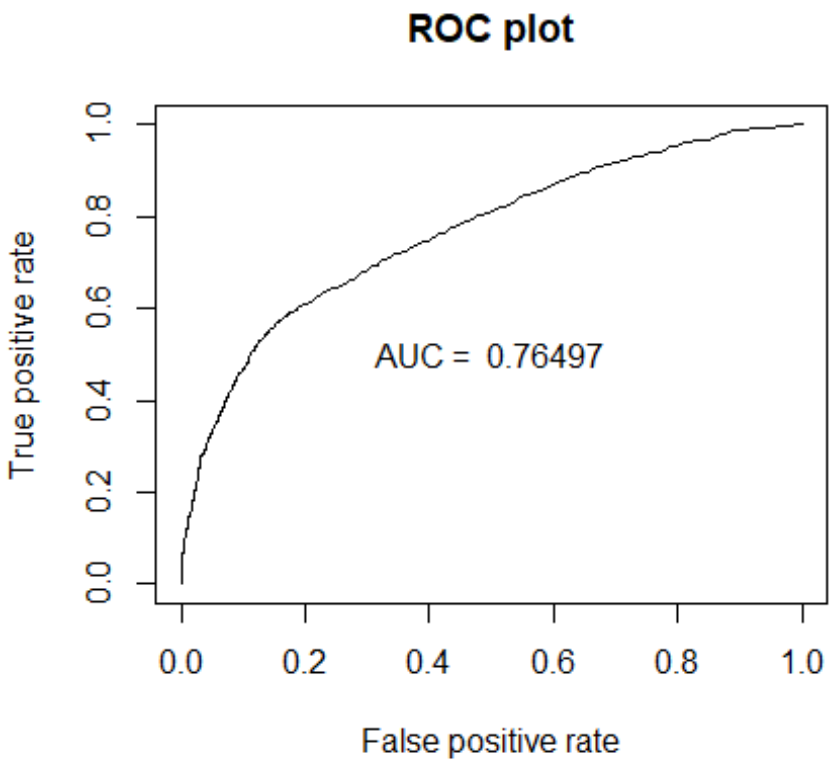

图 10 ROC 曲线

- 运行时间和内存消耗：建模过程中的时间占用和内存占用分别为 133.81s 和 4.61kb

4.3.2 AdaBoost 算法实证检验

本数据及采用上述介绍的数据集，将数据集随机分成训练集和测试集。采用分层抽样的方法，从原始数据中抽取 20%的数据作为测试集，剩下的 80%的数据作为训练集。分层抽样的好处在于能够保证训练集和测试集中因变量的取值分布与总体数据因变量的取值分布相同或者近似相同。

本例是一个典型的有监督机器学习算法，并且是简单的二分类算法。AdaBoost 能够将多个弱分类器进行线性组合，生成一个强分类器。AdaBoost 接受的自变量类型可以是分类型的数据，也可以是数值型数据。在 R 语言中，分类型数据被表示成因子型数据，因此，需要将原始数据中用数字表示的分类型标量转换成因子型数据。AdaBoost 的分类机制类似于逻辑回归，即首先计算出被分为 0 或 1 的概率，然后结果类别就是概率比较大的那个类。因此，AdaBoost 模型也会输出一系列的预测概率。

由于是二分类数据，我们经常采用混淆矩阵来表示模型的拟合效果。本例中，AdaBoost 的分类结果的混淆矩阵如下表所示。

表 5 混淆矩阵

-	-	预测	-	-
-	-	1	0	合计
实际	1	TP	FN	TP+FN
-	0	FP	TN	FP+TN
合计	-	**TP+FP**	**FN+TN**	**TP+FP+TN+FN**

其中，TP 表示 TRUE POSITIVE（真阳性），即实际结果为正，预测结果也为正的结果个数。FN 表示 FALSE NEGATIVE（假阴性），表示实际结果为正，但是预测结果为负的结果个数。FP 表示 FALSE POSITIVE（假阳性），表示实际结果为负，但是预测结果为正的结果个数。TN 表示 TRUE NEGATIVE（真阴性），表示实际结果为负，预测结果也为负的结果个数。

准确率、精确度和召回率用来评判模型的优劣程度。令$N = TP + FP + TN + FN$,则准确率的计算公式为：

$$Accuracy = \frac{TP + TN}{N}$$

精确度的计算公式为：

$$Precision = \frac{TP}{TP + FP}$$

召回率的计算公式为：

$$Recall = \frac{TP}{TP + FN}$$

F-Measure 是 Precision 和 Recall 加权调和平均

$$FMeasure = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times \frac{TP}{TP + FP} \times \frac{TP}{TP + FN}}{\frac{TP}{TP + FP} + \frac{TP}{TP + FN}}$$

$$= \frac{2 \times TP}{2 \times TP + FP + FN}$$

FMeasure 综合了 P 和 R 的结果，当 FMeasure 较高时则比较说明实验方法比较理想。

本例用于预测得出的混淆矩阵如下：

表 6 混淆矩阵

-	-	预测	-	-
-	-	1	0	合计
实际	1	447	880	1327
-	0	224	4449	4673
合计	-	**671**	**5329**	**6000**

则

$$Accuracy = \frac{TP + TN}{N} = \frac{447 + 4449}{6000} = 81.6\%$$

$$Precision = \frac{TP}{TP + FP} = \frac{447}{447 + 224} = 66.62\%$$

$$Recall = \frac{TP}{TP + FN} = \frac{447}{447 + 880} = 33.68\%$$

$$FMeasure = \frac{2 \times TP}{2 \times TP + FP + FN} = \frac{2 \times 447}{2 \times 447 + 224 + 880} = 44.74\%$$

从输出结果来看，虽然模型的准确率达到了 81.6%，但是其精确度却很低，召回率更是严重小于 50%。同时，FMeasure 的值也小于 50%，说明该模型还有待改进。

4.3.3 XGBoost 算法实证检验

4.3.3.1 XGBoost 实证检验

2015 年 8 月，XGBoost 的 R 包发布，将在本节引用 0.4-2 版本的 XGBoost 包。它类似于梯度上升框架，但是更加高效。它兼具线性模型求解器和树学习算法。最大的优势就是计算效率很高，它快速的秘诀在于算法在单机上也可以并行计算的能力。这使得 XGBoost 至少比现有的梯度上升实现有至少 10 倍的提升。 XGBoost 的 R 包对数据有一个限制就是仅适用于数值型向量。所以需要在使用中区分数据类型，比如“一年级”、“二年级”之类的分类变量，需要变成哑变量，然后进行后续的处理。并且 XGBoost 的空值是自己处理的，XGBoost 内置处理缺失值的规则。用户需要提供一个和其它样本不同的值，然后把它作为一个参数传进去，以此来作为缺失值的取值。XGBoost 在不同节点遇到缺失值时采用不同的处理方法，并且会学习未来遇到缺失值时的处理方法。因此在数据处理上就不用处理缺失值了。

模型建立

根据上一章所介绍的模型建立过程，而在 R 包中的 XGBoost 的建模函数中只需要进行简单的参数调整即可，学习率利用牛顿优化方方法会很大降低计算效率，因此在这里采用的是自己设定一个超参。

交叉验证确定迭代次数

利用训练集数据进行交叉验证分析。根据交叉验证所得到的测试均方误差和测试残差标准差，来选取较为合理决策树棵数即最佳迭代次数。交叉验证采取 10 折交叉验证，基学习器的深度规定为 2，学习率规定为 1，目标函数是基于二分类问题的 logistic 损失进行模型建立的，交叉验证结果如图所示：

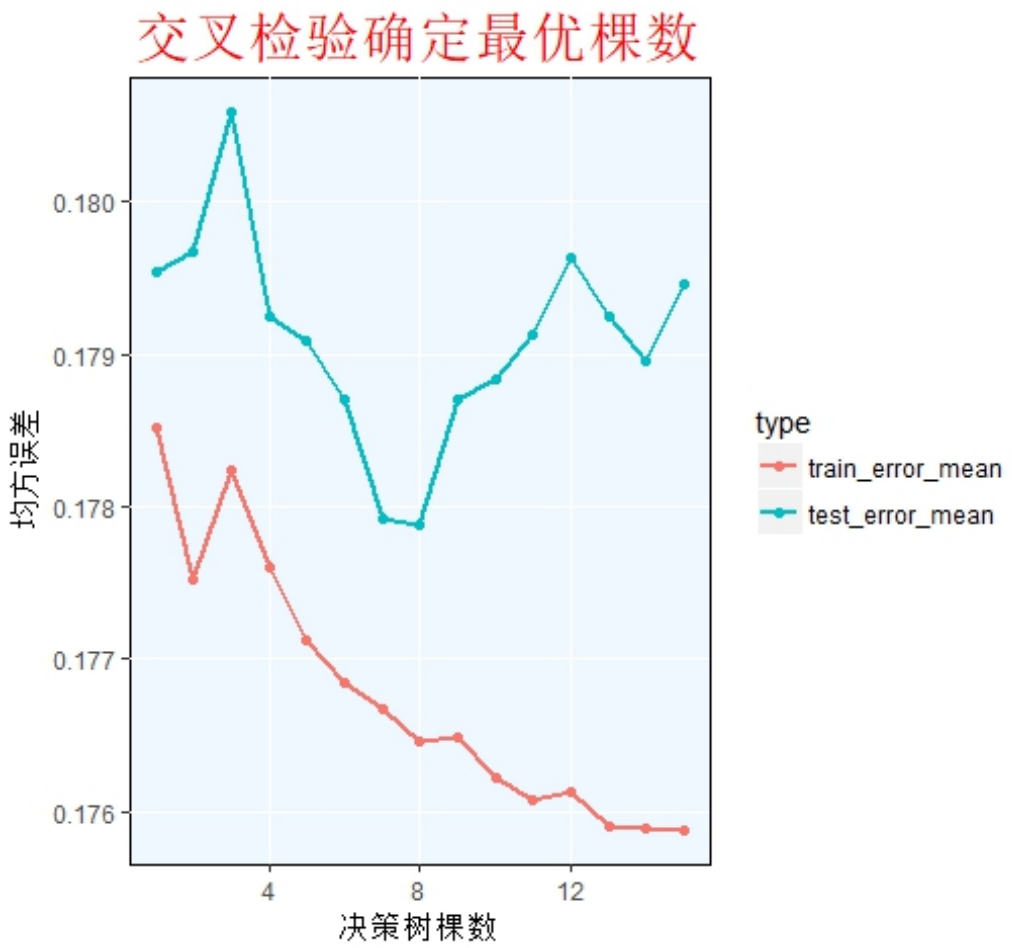

图11 交叉检验结果

从交叉验证图中都可以看出，训练均方误差在随着迭代次数的增加时一直在减少，而测试均方误差则是先减小，而后由于模型的复杂度的提升导致过拟合使测试均方误差增大。测试均方误差在迭代次数为 8 次时，XGBoost 模型在台湾信贷测试数据集上的表现达到最好。所以在 XGBoost 模型中选择 8 为迭代次数。

建立 XGBoost 模型

根据交叉验证结果，选择决策树的深度为 2，学习率为 1，迭代次数为 8 次的，建立 XGBoost 模型，设置的判断阈值为 0.5。由是单机并行处理，XGBoost 模型的训练时间很短，只有短短的 0.097s 相对于随机森林和 Adaboost 进步很多。模型的训练过程如下表 7 所示：

表 7 训练过程

训练次数	训练误差
1	**0.178500**
2	**0.178000**
3	**0.179375**
4	**0.178583**
5	**0.177208**
6	**0.178125**
7	**0.177875**
8	**0.177208**

通过模型的迭代次数的增加，模型的表现也在增加，均方误差在减小，XGBoost 模型的变量重要性如下图所示

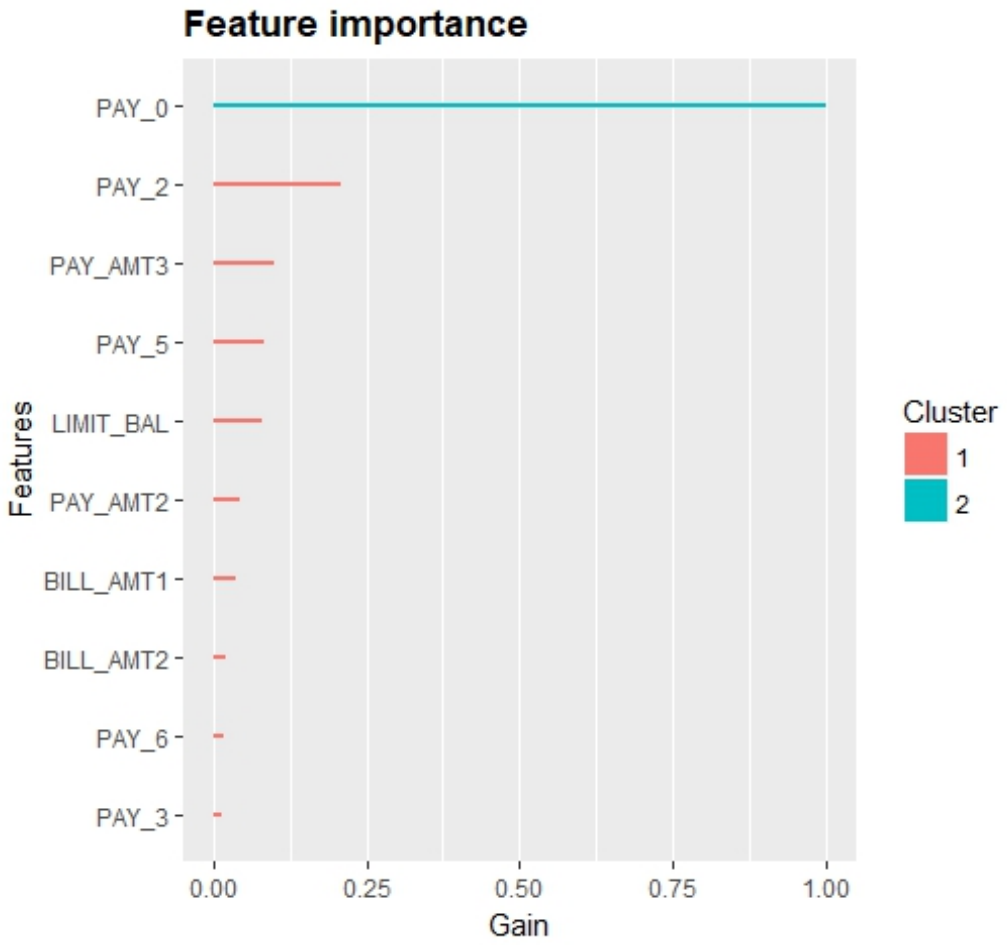

图 12 变量重要性

根据变量增益衡量的特征指标间的重要性衡量，可以看到，将变量大致分为了两类，其中第一类只有一个变量 PAY_0，代表的是前一年的还款情况，对系统的信息增益极大，其余的变量对系统的信息增益到比较小，都别划分为第二类。可以注意到还款行为是对 XGBoost 模型有着重要的影响作用。根据信息增益较高的变量生成的分类树如图 10 所示：

最终的 XGBoost 模型训练参数如下表所示：

表 8 评价指标

Accuracy	Recall	Precision	F_measure
0.8141667	**0.3409742**	**0.709389**	**0.4605709**

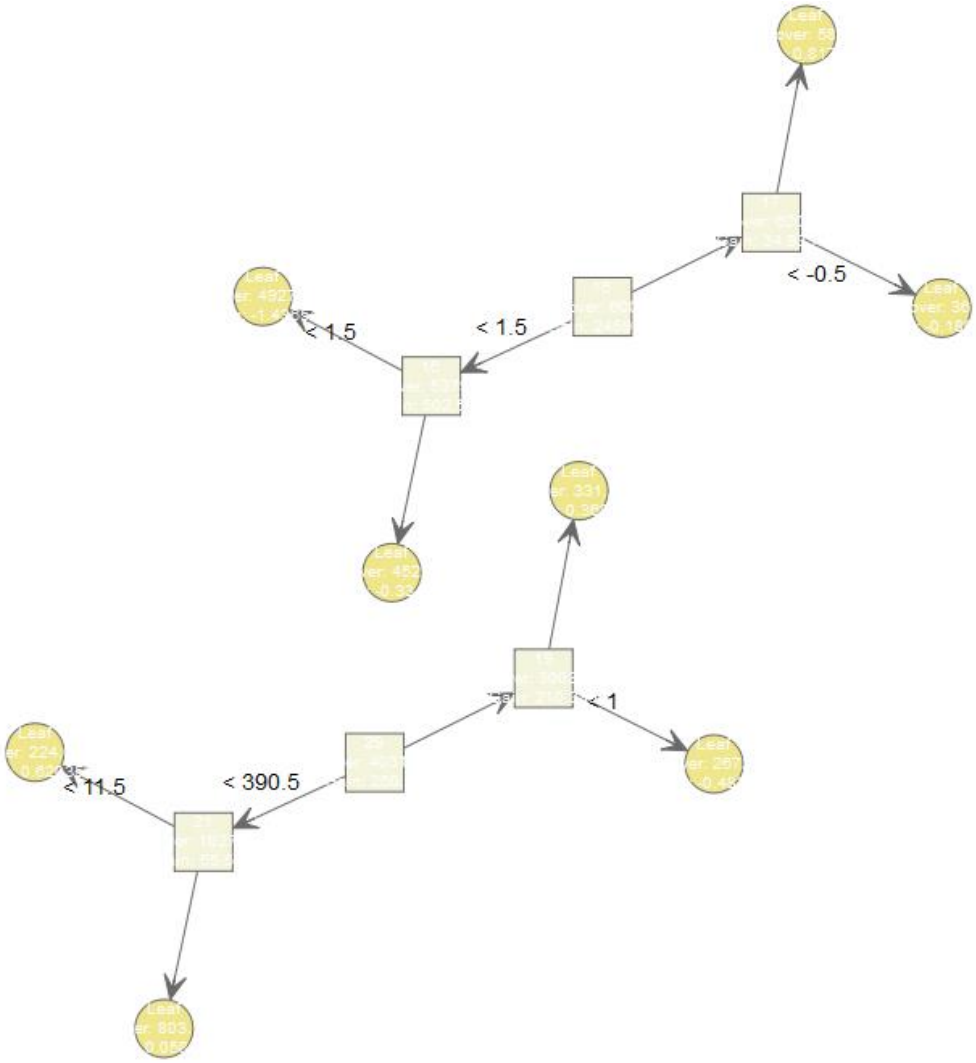

图 13 分类树示意图

在表 3-2 可以看到，模型的训练结果是比较不错的，总体的正确率已经达到了 81.4%，而且召回率也有了一定的提升 34.09%，准确率也达到 70.93%，但是 F 值仍旧没有超过 50%，比 Adaboost 和随机森林都有一定的提升。XGBoost 模型的分类混淆矩阵，如表 9 所示：

表 9 混淆矩阵

	Pred_0	**Pred_1**
True_0	4385(73.083%)	**219(3.650%)**
True_1	**900(15.000%)**	**496(8.267%)**

从表 3-3 中混淆矩阵可以看出 XGBoost 模型的正确率是很高的。已经可以将正确率提高到 82%左右，这个准确率比随机森林和 Adaboost 的结果已经好了，而且 XGBoost 的运算效率很高。但是由于分类器是会犯两种错误的，所以 XGBoost 的 18%的预测错误率是包含着将违约者判别到不违约，以及将不违约者分类到违约。由于数据是台湾的信贷数据，对于商业银行来讲，两种错误都会给银行带来损失，但相比之下，将违约者判断到不违约给银行造成的损失是比较大的。所以针对本次的信贷数据来讲，分类器的重点应该放在正确识别违约者上，提高真阳性率。为此课可以适当的提高假阳性率，也就是未违约者被分类到违约的错误率。 根据以上的思想，画出台湾信贷数据的 XGBoost 分类器的 ROC 曲线。如图 3-2 所示，AUC 取值为 0.773。ROC 曲线向左上角的偏移程度并不是很高，AUC 数值来说也并不是很高。

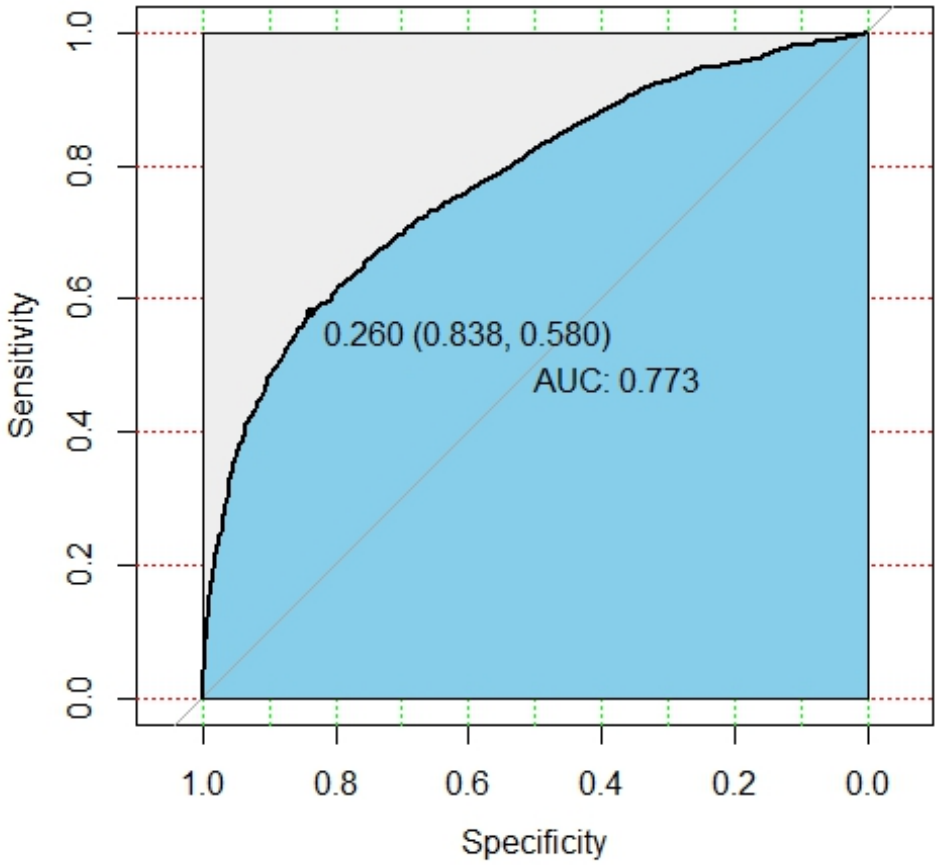

图 14 ROC 曲线

ROC 曲线中我们可以看到，由于此分类器处理的效果并不是很好，所以提高真阳性率的代价是将假阳性率也提高到较高程度。对于 0.5 的阈值可以做到的是将总的判断正确率提高到最高水平，但是违约正确识别率只有可怜的 35.3%。这是

给银行带来很大损失的。所以将判断阈值降低到 0.2，得到的混淆矩阵如下表 10 所示：

表 10 混淆矩阵

	Pred_0	**Pred_1**
True_0	3485(58.08%)	**1119(18.65%)**
True_1	**485(8.08%)**	**911(15.18%)**

建立的 XGBoost 模型的其他测试结果参数如下表 11 所示:

表 11 测试结果

Accuracy	**Recall**	**Precision**	**F_measure**
0.7271667	**0.6525788**	**0.4415899**	**0.5267418**

可以看到信贷 XGBoost 模型的总体准确率 accuracy 为 72.71%，仅比随机猜想高一点点。但是由于模型是为了信贷数据所搭建的，所以更重要的是 Recall 召回率，可以看到召回率已经达到 65.25%，已经可以区分出大约 2/3 的违约者了。但是准确率 precision 是较低的只有 44.15%，每判断一个违约者正确率不到 50%。由于 recall 与 precision 有时会出现矛盾，用 F_measure 即两者的加权调和平均衡量模型水平，F 值已经超过了 0.5 说明模型的总体的信贷违约率判别状况较好。在这一项的判别中已经比随机森林和 Adaboost 两种集成算法好的很多。XGBoost 模型就此建立完成。可以用来判断银行的识别违约用户。

模型进一步改进

根据图 3-3 的 XGBoost 模型的输出结果，可以知道每棵树将样本分类到哪片叶子上，facebook 最新发表的论文介绍过如何利用 GBDT 模型构造新特征的方法，并根据新构造的特征变量信息提高模型的表现。 论文的思想很简单，就是先用已有特征训练 GBDT 模型，然后利用 GBDT 模型学习到的树来构造新特征，最后把这些新特征加入原有特征一起训练模型。构造的新特征向量是取值 0/1 的，向量的每个元素对应于 GBDT 模型中树的叶子结点。当一个样本点通过某棵树最终落在这棵树的一个叶子结点上，那么在新特征向量中这个叶子结点对应的元素值为 1，而这棵树的其他叶子结点对应的元素值为 0。新特征向量的长度等于 GBDT 模型里所有树包含的叶子结点数之和。

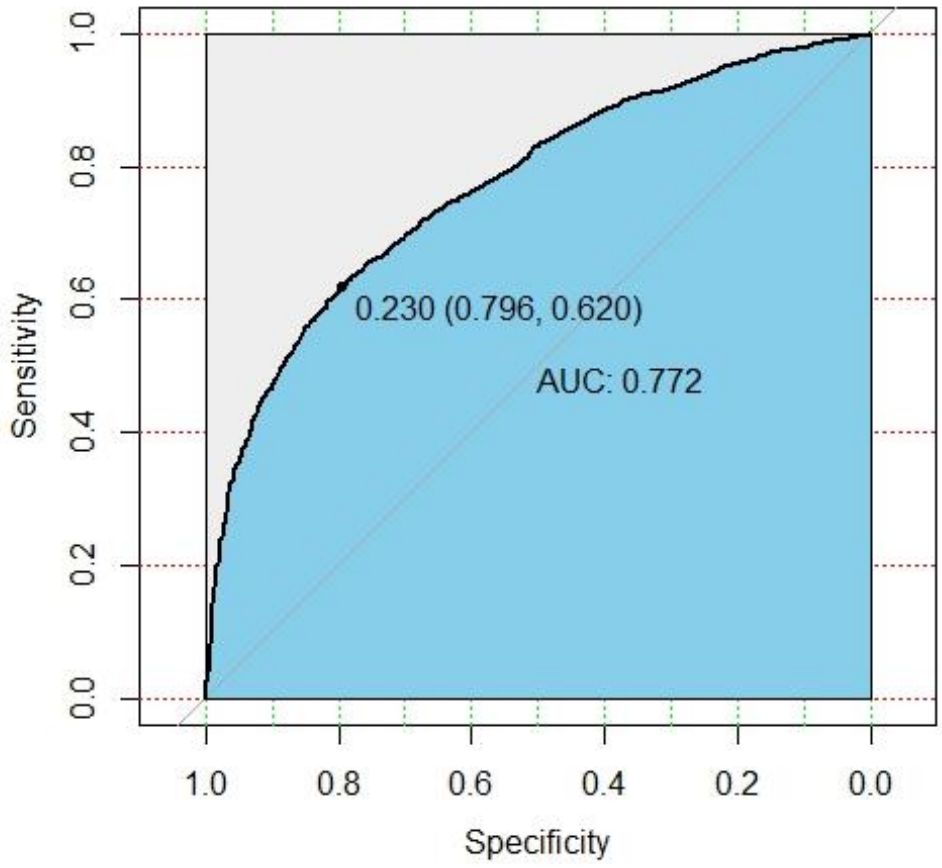

图 14 ROC 曲线

本节将以 XGBoost 的 GBDT 模型作为载体，尝试构造新的特征变量，用以改进和提高模型效果。新生成的叶子节点变量为共有 25 个，全部为 0-1 的稀疏矩阵，与原始变量进行合并后，进行新模型 XGBoost 的建模 经过十折交叉检验，设置的超参仍旧不变，均方误差仍为在 8 次迭代时最小，具有新特征变量的 XSGBoost 模型，仍旧观察模型的 ROC 曲线，如图 3-3，可以看到 ROC 曲线有了小幅改观，这已经实属不易，在集成算法中提升一丝都是逼近数据极限特征的努力。

增加新特征变量的 XGBoost 的模型，模型预测阈值仍取 0.2，所得到的性能结果如下表 12 所示：

表 12 训练结果

Accuracy	Recall	Precision	F_measure
0.7273333	**0.6604585**	**0.4424184**	**0.5298851**

可以看到信贷 XGBoost 模型的总体准确率 accuracy 提升为 72.73%，较原始模型提升了 0.5%，但是由于模型是为了信贷数据所搭建的，所以更重要的是

Recall 召回率，可以看到召回率已经达到 66.05%，已经可以区分出 2/3 的违约者了。准确率 precision 也提高到 44.24%， F 值提升了 0.7%。增加叶子特征变量的 XGBoost 模型就此建立完成。由于 XGBoost 模型在迭代 8 次后已经出现过拟合状况，所以增加新特征变量效果已经不大，但还是有所提升，所以可以用来判断银行的识别违约用户。

4.4 结果与比较

先不考虑信贷的实际问题，单纯比较算法性能来讲，集成算法的对比表如下表所示，可以看出单从准确率上看，算法之间差距并不是很大，说明在集成算法上可能都已经逼近了数据本身应有的特征，无法有大的改进，但值得注意的是，随机森林和 Adaboost 都需要进行多棵决策树的生成，迭代次数较多，成千上百次，而 XGBoost 只需要进行少量的迭代次数，在本次试验中仅需要 8 次。而且运行时间上 XGBoost 的多线程并行运行，也发挥了极大作用，比 Adaboost 缩短时间达到了近 100 倍，而比随机森林缩短时间近 200 倍。在本次信贷的数据分析中，XGBoost 更有调节阈值后可以更加符合银行的商业要求，更加准确的识别出违约人。

表 13 算法比较

	随机森林	Adaboost	XGBoost	改进 XGBoost
Accuracy	0.81	0.82	0.83	**0.82**
recall	0.37	0.34	0.36	**0.36**
precision	0.62	0.67	0.68	**0.68**
F_measure	0.46	0.45	0.48	**0.49**
运行时间(s)	**44.46**	**23.55**	**0.21**	**0.23**

五、结论与展望

5.1 结论

集成学习算法通过建立几个模型组合的来解决单一预测问题，无论是 boosting 还是 bagging 的思路表现均不错。在本文针对台湾信贷数据的分类预测中，三种集成分类器预测准确率均在 80%左右；随森林模型的最终 F 值为 0.465，XGBoost 模型在优化前后的 F 值分别为 0.527 和 0.530，Adaboost 的 F 值为 0.447；在 F 值上 XGBoost 模型表现要比其两个模型优越。针对信贷数据对精确度比较敏感的特点，对 XGBoost 阈值的调整有些许提高，但以降低预测准确度为代价。综上所述，集成学习的三个典型算法在针对多类型特征的二分类预测上表现良好，但是普遍的准确度均较低。

5.2 不足与展望

本文对三个典型的集成算法在台湾信贷数据的应用效果进行了对比，除了比较boosting 和 bagging 两种思路对应的算法特点，还对 boosting 中具体细分原理的不同算法进行了对比；相比过去算法比较时一般只比较预测效果还对比了三种不同算法的内存占用和时间消耗，为相关学习者提供了更多参考。 尽管尝试了多种参数对算法进行调试，但本文对三种集成算法的改进任然不够，特别是对于弱分类器的种类均基于决策树。未来进一步工作将尝试更多的弱分类器种类基础上的集成算法的效果比较和模型调试。

参考文献

[1]李航.统计学习方法[M].北京：清华大学出版社,2012.

[2]Tianqi Chen,Carlos Guestrin.XGBoost:A Scalable Tree Boosting System[EB/OL].http:arxiv.orgabs1603.02754,2016.

[3]曹莹,苗启广,刘家辰,高琳.AdaBoost 算法研究进展与展望[J].自动化学报,2013,39(06):745-758.

[4]方匡南,吴见彬,朱建平,谢邦昌.随机森林方法研究综述[J].统计与信息论坛,2011,26(03):32-38.

[5]冯少荣.决策树算法的研究与改进[J].厦门大学学报(自然科学版),2007(04):496-500.

[6]李旭升. 贝叶斯网络分类模型研究及其在信用评估中的应用[D].西南交通大学,2007.

[7]Sun J,Li H,Listed companies financial distress prediction based on weighted majority voting combination of multiple classifiers. Expert Systems with Applictions.2008,35(3):818-827

[8]West.D,S Dellana,J.Qian. Neural network ensemble strat-egies for financial decision applications. Computers&Oper-ations Research . 2005

[9]Finlay,S.H.Towards Profitability:A Utility Approach to The Credit Scoring Problem. Journal of the Operational Research Society . 2008

[10]Finlay S M.Credit scoring for profitability objectives. European Journal of Operational Research . 2010