数据挖掘模型在小企业主信用评分领域的应用

国家统计局国际统计信息中心　王磊、范超、解明明

摘要：中国金融改革正引导商业银行做好小企业（主）贷款业务，使更多信贷资源流向实体经济。作为一种国际经验，信用评分技术可在很大程度上解决小企业贷款高成本、高风险及信息不对称难题。

本文广泛选取了可能适用于小企业主信用评分领域的 12 个数据挖掘模型（包括本文对 Logistic 模型的改进模型-门限 Logistic 模型），并以实际案例，通过 10 折交叉验证和预期分类错误成本的方式，检验了这 12 个模型的信用评分能力。分析结果表明，本文改进的门限 Logistic 模型在模型预测能力、稳定性、犯错率及预期错误分类成本等诸多方面均表现优秀；而 Boosting 在预期错误分类成本上的表现仅次于门限 Logistic 模型，显示出较强的信用评分能力，具有较好的推广性。

本文研究对国内商业银行建立合适的小企业主贷款信用评分模型具有参考意义；而该类模型的实施预期可推动商业银行进一步完善微观金融数据统计工作，进而有助于推动宏观政府金融统计工作。

引言

中国金融改革正引导商业银行做好小企业（主）贷款业务，使更多信贷资源流向实体经济。然而，小企业贷款具有单笔贷款金额小、笔数多、总成本高、信息不对称等问题。作为一种国际经验，信用评分技术的引入可在很大程度上解决上述难题。一方面，信用评分模型能够简化贷款手续,加快信贷决策速度,降低小企业信贷的交易成本。另一方面，小企业信用评分体系有利于解决信息不对称引起的逆向选择问题和道德风险问题（向晖，2011）。

小企业主信用评分是指使用某种数量分析方法,综合考虑影响小企业主(及其家庭)信用的主客观环境,对其履行各种经济承诺的能力进行全面的判断和评估,并以一定的数值表明其信用状况。

小企业主信用评分模型的构建及其未来在国内商业银行较大范围的应用,不论对于微观层面国内商业银行做好小企业贷款业务,还是对于宏观层面推动实体经济进而整个中国经济持续发展均具有重要的现实意义：（1）Allen N. Berger（2005)的研究表明小企业信用评分模型的使用可节省银行近50%的成本，随着国内银行在小企业（主）信贷方面经验和数据的积累，小企业主信用评分模型在节约成本、提高效率、客观公正等方面的优势，必将为各银行所青睐。（2）宏观层面，Frame、Srinivasan 和Woosley（2001）估计1997 年大银行使用信用评分后，低于10 万美元小企业信贷资产池将增加8.4%；而Allen N. Berger（2005）的研究发现小企业信用评分的使用与小企业信贷（10 万美元）的增加有很大关系。因此，小企业主信用评分模型在国内商业银行的使用预期可提高小企业（主）的贷款可得性，进而为实体经济的持续繁荣和中国经济的健康发展提供必要推动力。

而本文的主要目的，就是通过比较可能适用于小企业主信用评分领域的12个模型，从中揭示出各类模型的特性及在该领域的评分能力表现,以期为国内商业银行未来做好小企业主信用评分模型建立和实施提供一定的参考。

一、文献回顾

从现有文献来看，一些学者很早就开始关注不同信用评分方法的应用效果比较。Myersand 和 Forgy (1963)比较了多元线性回归和判别分析方法在识别个人信用风险方面的能力，结果表明判别方法略优于线性回归。wiginton (1980)认为 Logistic 回归能够取得比判别分析更好的效果。Grablowsky 和 Talley (1981)则认为 Probit 回归能够比多元判别分析方法更好的鉴别个人信用风险。

Hardy 和 Adrian（1985)的研究表明线性规划比判别分析法效果更好，而且能够根据信用政策进行灵活的调整。Coffman（1986)认为决策树和判别分析应用于信用评分的效果跟特征变量间的相互作用有关。Nath 和 Jones (1992)的研究表明线性规划在信用评分方面的应用效果不如统计模型。

Boyle 和 Haniilton(1992)认为决策树比判别分析更适用于个人信用评分。Davis 和 Edelman 等（1992)将 ID3、CART 决策树方法和多种神经网络方法应用于个人信用评分的效果进行了比较，结果表明各种方法的分类精度接近。Altman 和 Marco (1994)研究了多元判别分析和神经网络在识别信用风险方面能力,结果表明神经网络的预测精度明显高于多元判别分析。

Desai 和 Convay 等（1997)在信用局环境下将前馈神经网络、遗传算法跟 Logistic 回归和判别分析方法在预测个人信用风险方面的能力进行了比较,结果表明人工智能方法比统计方法能够更好的模拟变量间的非线性关系。Arminger 和 Enache (1997)比较了决策树、神经网络以及 Logistic 回归方法在预测违约贷款方面的能力,结果表明三种方法的效果相当。

Piramuthu (1999)利用了三个信用数据库对多层神经网络以及模糊神经网络进行了比较，结果表明模糊神经网络具有更好的可解释性。

West(2000)将五种神经网络模型与线性判别分析、Logistic 回归、K 最近邻、决策树等模型在信用评分中的预测精度进行了比较，结果表明 MOE 和 RBF 神经网络优于 MLP 神经网络，同时 Logistic 回归模型在所有统计模型中预测精度最高。

石庆族和靳云汇（2004)在中国个人信用数据上比较了判别分析、Logistic 回归、线性规划、神经网络及决策树方法的应用效果，结果表明神经网络预测精度高于其他模型，而 Logistic 回归具有更低的第二类错误率且更稳健，因此 Logistic 回归模型的综合性能最优。

Lee 和 Chen(2005)与 Akkoç （2012）分别提出了 MARS 与 BPN 神经网络组合的两阶段混合模型及三阶段的混合自适应神经模糊推理系统，并认为各自的模型在平均正确分类率与预期误判成本方面均优于线性判别分析、Logistic 回归分析和人工神经网络（ANN）等模型。

从以上比较研究来看，学者们对各种评分方法的效果并未达成共识，甚至还互相矛盾。这种现象的出现主要有三方面原因：第一，研究者们采用不同的个人信用数据对各种方法进行比较，显然各种建模方法应用于不同数据库时性能是不一样的。第二，研究者们采用的各种建模方法的结构、参数设定有差异。而神经网络、支持向量机、聚类等方法对模型结构和参数非常敏感。第三，绝大多数研究者们未公布他们的的数据预处理和指标体系选择方法和结果，这也极大影响了对不同模型之间的比较研究结果。

综上，现有文献在小企业信用评分模型比较方面已做了较充足的研究。根据现有研究，在诸多常用方法中，Logistic 回归在稳健性上表现较为突出，部分文献也认为其预测精度较优甚至最高（如 West，2000；石庆族和靳云汇，2004），且 Logistic 回归具有很好的可解释性。考虑到国内银行业在小企业主信用评分领域尚处在初步阶段，因此以该模型为研究基础较为合适。不过小企业信用评分常用的 Logistic 回归模型仍然需要改进。在某一特征（例如小企业主家庭负债率）上有显著差别的小企业可能在违约规律上有显著差别，而现有研究没有充分考虑这一问题。为此，本文拟引入门限模型对现有 Logistic 回归模型进行改进，以期提高小微企业评分模型的预测准确率和降低预期误判成本，同时降低商业银行小微贷款审核工作负担，进而提高其实际应用价值。

更为重要的是，为了做出综合而客观的比较结论，除了本文改进模型-Logistic 门限模型外，本文还广泛选取了可能适用于小企业主信用评分领域的其余 11 个模型，从模型驱动角度来对共计 12 个潜在模型的信用评分性能做了全面、客观和严谨的比较。

余文安排如下：第二部分简要介绍 12 个模型；第三部分进行数据描述；第四部分开展实证研究；第五部分对模型的信用评分能力做出比较和评价；第六部分做结论和研究展望。

二、模型

本文广泛选取了可能适用于信用评分领域的 12 个模型，涵盖算法模型和数据模型中较常用模型，篇幅所限，下文仅简要介绍各模型基本思想。

（一）基于树类的模型

1.决策树（分类树）

Makowski (1985)和 Lee (2006)等的研究认为决策树方法在信用评分中表现优良。决策树方法具有自动选择变量、较好地处理缺失信息、准确性较高等优点，曾被美联储在《平等贷款机会法》中称为在信用系统中在统计意义上完美的办法。决策树学习是应用最广的归纳推理算法之一。它是一种逼近离散值函数的方法，对噪声有很好的健壮性且能够学习析取表达式。决策树一般都是自上而下来生成的，并用了贪婪的搜索遍历方法进行遍历。决策树通过把实例从根节点排列到某个叶子节点来分类实例，叶子节点即为实例所属的分类，树上的

每个节点指定了对实例的某个属性的测试，并且每个节点的每一个后继分支对应于该属性的一个可能的值。

2. 组合模型：Bagging、随机森林和 Boosting 模型

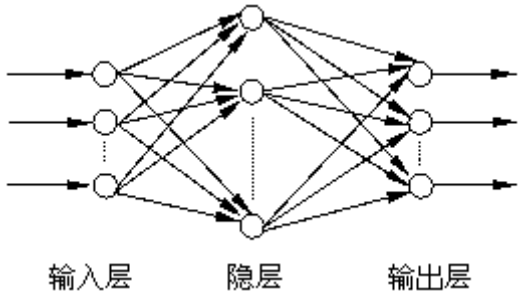
Bagging（bootstrap aggregating 的缩写）算法是最早的集成学习算法。它也是最具有指导意义和实施最简单，而且效果惊人的好的集成学习算法。Bagging 算法的多样性是通过由有放回抽取训练样本来实现的，用这种方式随机产生多个训练数据的子集，在每一个训练集的子集上训练一个分类器，最终分类结果是由多个分类器的分类结果多数投票而产生的。虽然这个算法很简单，但是当这种方法集成基础学习器的泛化策略可以降低偏差。Bagging 算法当可用的数据量有限时结果更加吸引人。

随机森林与 Bagging 的区别在于在每个分割节点处是随机选取一定数量的变量，避免某些“强势”变量支配模型。Bagging 可看做是随机森林的一种特殊情况，即若在每个分割节点处选取了全部变量，此时随机森林即为 Bagging 模型。

Boosting 算法包含了一系列方法。与 Bagging 不同，Boosting 算法通过顺序给训练集中的数据项重新加权创造不同的基础学习器。Boosting 算法的核心思想是重复应用一个基础学习器来修改训练数据集，这样在预定数量的迭代下可以产生一系列的基础学习器。在训练开始，所有的数据项都被初始化为同一个权重，在这次初始化之后，每次增强的迭代都会生成一个适应加权之后的训练数据集的基础学习器。每一次迭代的错误率都会计算出来，而且正确划分的数据项的权重会被降低，然后错误划分的数据项权重将会增大。

（二）BP 人工神经网络

BP（Back Propagation）网络由 Rumelhart 和 McCelland 为首的科学家小组在 1986 年提出，Odom (1990)首次将神经网络方法引入信用风险评估。BP 网络是一种按误差逆传播算法训练的多层前馈网络，是目前应用最广泛的神经网络模型之一。BP 网络能学习和存贮大量的输入-输出模式映射关系，而无需事前揭示描述这种映射关系的数学方程。它的学习规则是使用最速下降法，通过反向传播来不断调整网络的权值和阈值，使网络的误差平方和最小。BP 神经网络模型拓扑结构包括输入层（input）、隐层(hide layer)和输出层(output layer)（如下图所示）。



图表 1 人工神经网络示意图

（三）支持向量机（SVM）

Baesens 和 Gestel (2003)最早将支持向量机方法运用于信用评分领域。支持向量机方法是建立在统计学习理论的 VC 维理论和结构风险最小原理基础上的，根据有限的样本信息在模型的复杂性（即对特定训练样本的学习精度）和学习能力（即无错误地识别任意样本的能力）之间寻求最佳折衷，以期获得最好的推广能力。支持向量机是从线性可分情况下的最优分类面提出的，它是实现统计学习理论思想的方法。 所谓最优分类面就是要求分类面不

但能将两类无错误地分开，而且要使两类的分类间隔最大。前者是保证经验风险最小（如使训练误差为 0），而使分类间隔最大实际上就是使推广性的界中的置信范围最小，从而使真实风险最小。

（四）K 近邻

Chatterjee 和 Barcun(1970)最先将最近邻法应用于建立个人信用评分模型。K 近邻(k-Nearest Neighbour，KNN)分类算法的思路是：如果一个样本在特征空间中的 k 个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。KNN 算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。KNN 算法不仅可以用于分类，还可以用于回归。

（五）线性判别分析（LDA）和二次判别分析(QDA)

Fisher (1936)开创了 fisher 判别分析，David Durand(1941)最早将该法应用于识别不良贷款，之后 Eisenbeis (1977,1978)则作了应用推广。本文线性判别分析选取的是贝叶斯判别分析。贝叶斯的思想总是假定对所研究的对象已有一定的认识,常用先验概率分布来描述这种认识 。然后我们抽取一个样本,用样本统计来修正已有的认识(先验概率分布)，得到后验概率分布。 各种统计推断都通过后验概率分布来进行，将贝叶斯思想用于判别分析就得到贝叶斯判别法。当不同类样本的协方差矩阵相同时，即是线性判别分析（LDA）的情况；当不同类样本的协方差矩阵不同时，则应使用二次判别分析(QDA)。

（六）Logistic和Probit回归

Wiginton(1980)和 Grablowsky 和 Talley (1981)分别最先将 Logistic 回归和 Probit 回归用于信用评分。用 Logistic 回归建立信用评分模型的目的就在于试图用下面的表达式来估计一个借款客户是好客户（y=0）或者坏客户(y=1)的概率：

$$p(y=1|X)=\frac{\exp(\beta_0+\beta^T X)}{1+\exp(\beta_0+\beta^T X)} \qquad (1)$$

式中，X 是 m 维自变量观测值矩阵，β 是 m 维待估计的参数向量。Logistic 回归通过最大似然估计方法求解回归参数。Probit 模型与 Logistic 模型类似，区别仅在于在 P（Y=1）=f(X)这一表达式中 f（.)是正态分布函数，而 Logistic 是 Logistic 函数。

（七）门限Logistic回归

本文认为小企业主的违约规律可能存在门限效应，即在某一特征（如家庭负债率）上有显著差别的不同群体其违约规律也存在显著差别。为此，本文引入门限效应对现有 Logistic 回归模型进行改进。作为一种初步尝试，本文仅讨论了单门限模型的情况。

门限模型是通过确定回归方程中门限变量的取值，将方程划分在不同的区间中，而每一个区间由不同的回归方程来表达。本文中门限值按照“分类总正确率”最大化的原则来确定。考虑“家庭负债率”对小企业主/个人违约行为产生影响的门限效应后，式（1）变为：

$$p(y=1|X)=\frac{\exp(\beta_0+\beta^T X)}{1+\exp(\beta_0+\beta^T X)}I(q_i\le\gamma)+\frac{\exp(\beta_0'+\beta'^T X)}{1+\exp(\beta_0'+\beta'^T X)}I(q_i>\gamma)+e_i \qquad (2)$$

上式中，X 含义同式 1；$1\le i\le n$，n 为样本总数；β为待估计参数；$I(\cdot)$ 为示性函数，即条件成立时取 1，否则取 0；q_i 为门限变量，e_i 为随机误差项，假定 $e_{it}\sim iid(0,\sigma^2)$ 。

门限Logistic模型在每一个区间依然使用普通Logistic方法估计参数。我们用 $RP_1(\gamma)$ 来表示 $q_i = \gamma$ 时门限 Logistic 模型得到的总分类正确率。所要确定的门限值 γ 是使 $RP_1(\gamma)$ 取得最大值时的 γ，即

$$\hat{\gamma} = \arg\max_{\gamma} RP_1(\gamma) \tag{3}$$

显然，门限变量 q_i 至多有 n 种取值，通过遍历搜索 q_i 各个取值即可确定出 $\hat{\gamma}$。若 n 较大，则采取格点搜索的方式来确定 $\hat{\gamma}$。当求出 $\hat{\gamma}$ 后，可得到最终的 $\hat{\beta} = \hat{\beta}(\hat{\gamma})$。

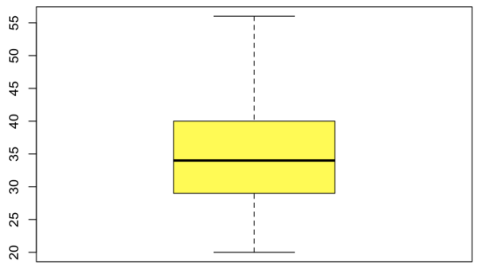
三、数据描述

本文数据来自 SPSS 软件自带的“bankloan.sav”，是某银行客户违约信息。本文判定的对象是银行客户是否违约（是否拖欠贷款），样本量为 700。其中，违约的样本量为 183，约占总样本量的 26%，非违约的样本量为 513，约占总样本量的 76%。选取的有关自变量为：年龄、受教育年限、现职业从业年限、本地生活年限、家庭收入、负债/家庭收入、信用卡负责和其他债务。因变量和自变量的说明如下表所示。

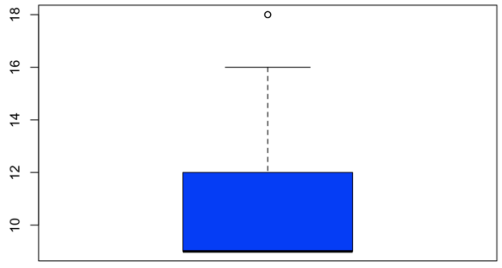
表格 1 因变量和自变量说明

变量符号	变量名称	类型	单位/说明
y	是否违约	定性变量	y=1 违约；y=0 非违约
x1	年龄	连续变量	岁
x2	受教育年限	连续变量	年
x3	现职业从业年限	连续变量	年
x4	本地生活年限	连续变量	年
x5	家庭收入	连续变量	千美元
x6	负债/家庭收入	连续变量	%
x7	信用卡负债	连续变量	千美元
x8	其他债务	连续变量	千美元

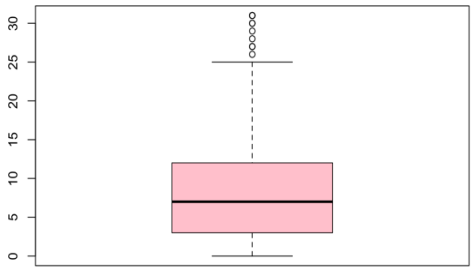
自变量的描述统计箱线图如下所示。



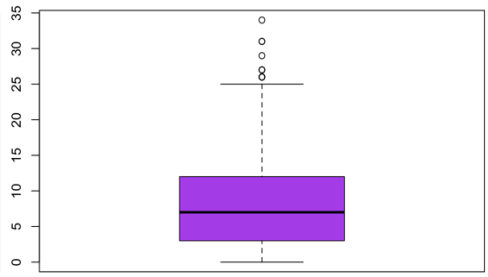
x1 年龄



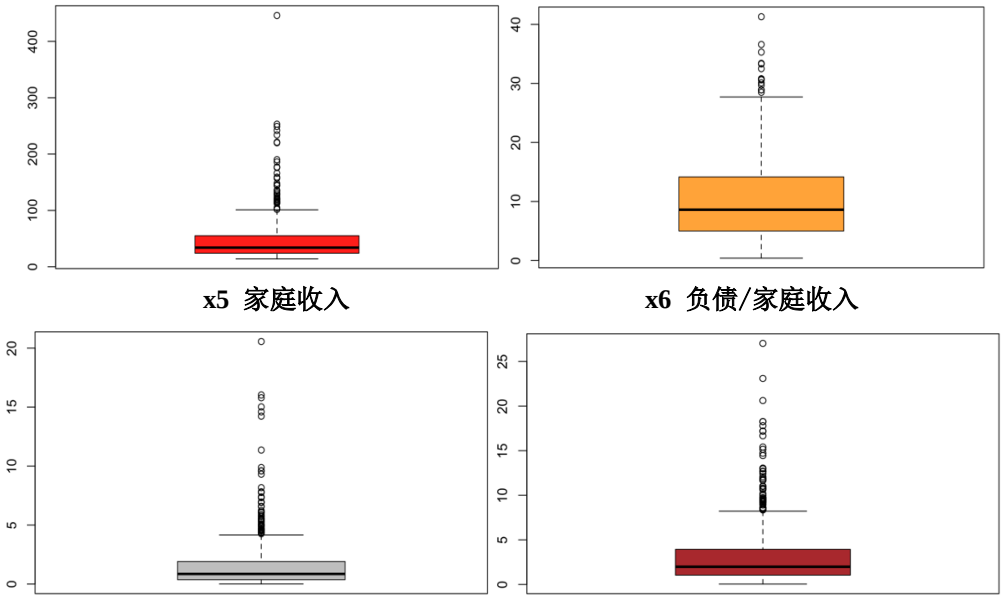
x2 受教育年限



x3 现职业从业年限



x4 本地生活年限



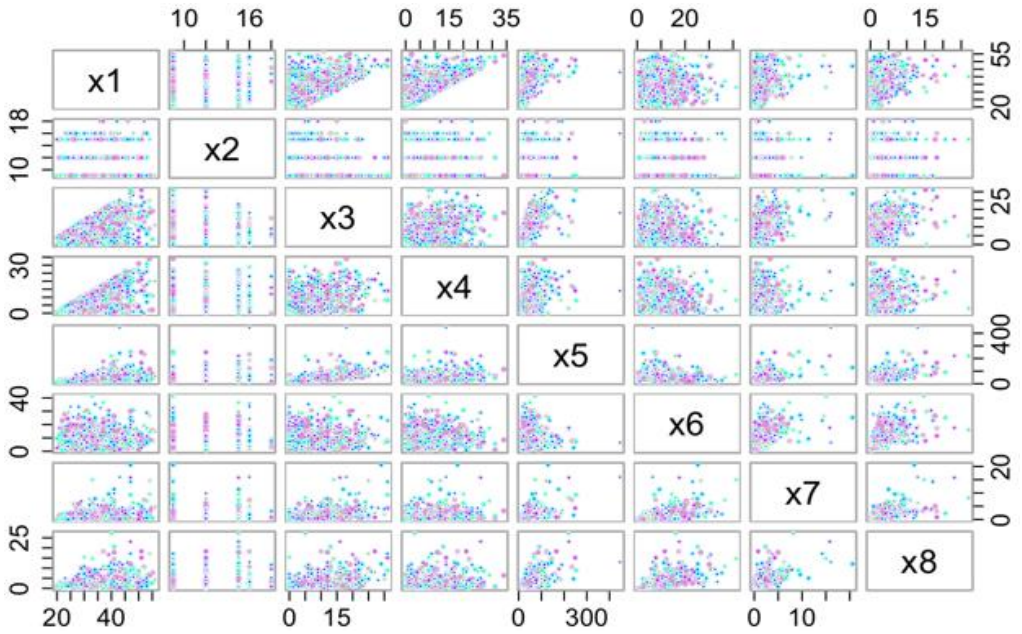
图表 2 自变量的箱线图

由上图可知，x1（年龄）分布较接近正态分布，其余自变量分布都呈偏态，其中 x5、x7、x8 偏态性非常强。x2（受教育年限）分布普遍集中在 9 至 12 年，有一个异常值。x3（现职业从业年限）分布从 0 到 31 年，中间百分之 50%的人工作 3 年到 12 年，有 6 个异常值。x4（本地生活年限） 分布与 x3 相似，中间百分之 50%的人在本地生活在 3 至 12 年，有 5 个异常值。x6（家庭负债率）有较多的温和异常值，而 x5（家庭收入）、x7（信用卡负债）、x8（其他债务）分布右偏，中间 50%数据非常集中，而较大数据分布非常分散。

变量间的相关系数表及图，如下所示。x6（债务/家庭收入）、

表格 2 自变量间的相关系数表

变量	x1	x2	x3	x4	x5	x6	x7	x8
x1	1.00	0.02	0.54	0.60	0.48	0.02	0.30	0.34
x2		1.00	-0.15	0.06	0.23	0.01	0.09	0.17
x3			1.00	0.32	0.62	-0.03	0.40	0.41
x4				1.00	0.32	0.01	0.21	0.23
x5					1.00	-0.03	0.57	0.61
x6						1.00	0.50	0.58
x7							1.00	0.63
x8								1.00



图表 3 自变量相关系数图

总体而言，本文 8 个自变量间的相关性不强 。除 x3 与 x5、x5 与 x8、x7 与 x8 间的相关系数超过 0.6，其余自变量间的相关系数均小于 0.6，且绝大部分小于 0.5。x2 与 x3，x3 与 x6，x5 与 x6 间呈非常弱的负相关，其余自变量间呈较弱的正相关。

四、实证研究：模型参数选择及对全样本的分析

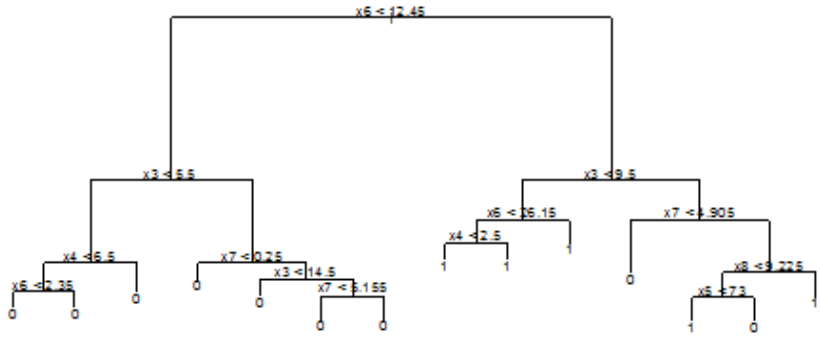
算法模型需确定一些参数，如组合模型中树的棵树，人工神经网络隐含层的层数，K 近邻中 K 值等。本部分随机将 700 个观测值分为训练集和测试集相等的两部分，通过观察不同参数下模型正确率的变化情况，以确定出模型的最优参数，为下一部分 10 折交叉验证等做准备。同时，在确定出各模型的参数后，各模型对全样本（即全部 700 个观测值进入模型）进行计算，求出相应的分类正确率，找出最重要的解释变量，对模型进行初步的比较分析。

本文使用 R 软件和 Matlab 软件获确实证结果（在 10 折交叉验证中使用了同一组随机数，故结果间完全可比）。程序为自行编写（如门限 Logistic 程序）或在 R 现有程序包基础上修改。主要程序请见附件 2。

1、基于树类的模型结果

（1）决策树（分类树）结果

我们首先使用分类树对全样本（即全部 700 个观测值进入模型）进行计算，决策树的输出结果见附件 1，相应的分类树如下图所示。

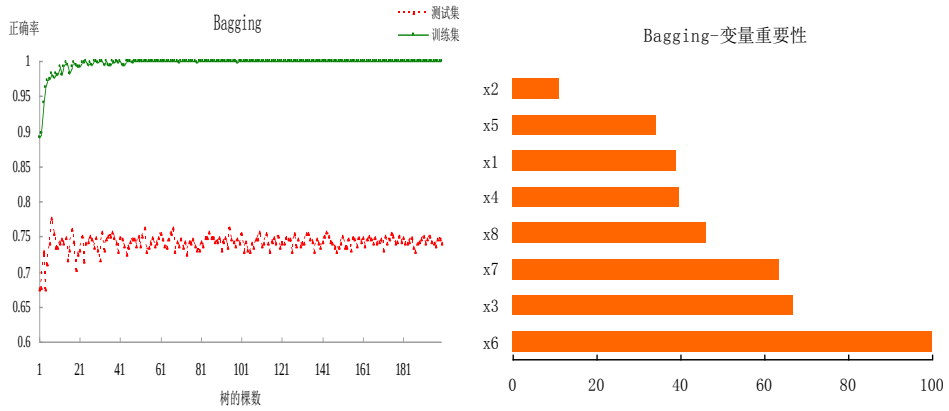


图表 4 决策树结果

由分类树的结果可知，在选取的 8 个变量中，变量 x6（负债/家庭收入)对于判定违约情况起了决定作用，当负债/家庭收入小于 12.45 时，全部判定为非违约，而当负债/家庭收入大于等于 12.45 时，绝大部分情况被判定为违约。分类树的整体正确率为 80.3%。

（2）Bagging 模型结果

在 Bagging 模型中，需要确定合适的树的棵数，棵数过少会造成拟合不充分，过多则有可能造成过拟合从而大幅加大模型方差。在确定树的棵数时，我们首先将样本随机分为训练集和测试集两部分，样本量各占 50%，分别计算棵数从 1 课到 200 棵相应的训练集和测试集分类正确率，最后通过正确率的变化情况确定树的棵数，变化情况如下图（左）所示。



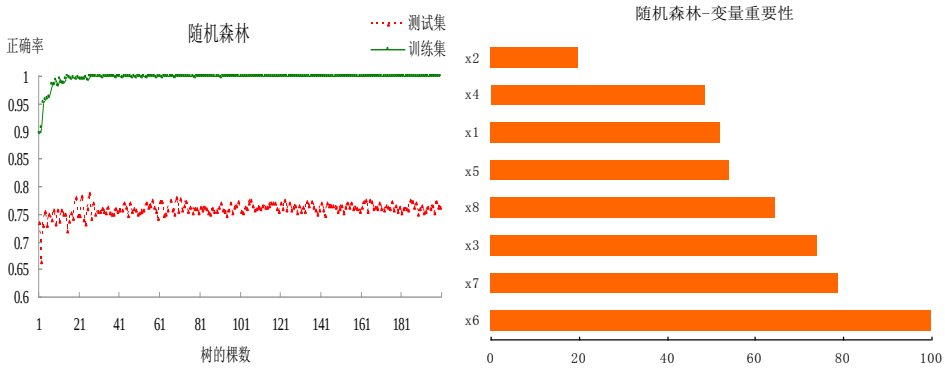
图表 5 Bagging 模型正确率随棵数变化情况（左）及变量重要性（右）

由上图（左）可知，对于测试集，树的棵数从 1 到 10 棵时，正确率从 0.89 迅速上升到 0.98 左右，当树的棵数达到 100 棵后正确率已稳定在 1；对于测试集，当树的棵数从 1 到 6 时正确率迅速上升，之后基本在 0.75 上下波动，当棵数达到 160 棵以后正确率趋于稳定，且未出现过拟合迹象。因此，在计算中，树的棵数选取大于 160 棵即可（本文使用了软件包默认值 500 棵）。

虽然在组合模型中不能像单棵分类那样给出明确的分类规则，但可以得到每个变量的相对重要性。在组合模型中，我们可以计算对于所有的树，平均而言，某个变量使得基尼系统下降的总和。该值越大，表示该变量在组合模型中越重要。我们对全样本做 Bagging，然后将变量重要性做标准化处理，使最大值为 100，求出其他变量重要性的相对值，其结果见上图（右）。可知，在 8 个变量中，最为重要的是 x6（负债/家庭收入），最为不重要的是 x2（受教育年限），相对较为重要的是 x3（现职业从业年限）和 x7（信用卡负债 ）。该结果与分类树的结果具有一定的一致性。

（3）随机森林结果

与 Bagging 法类似，我们将样本随机分为训练集和测试集两部分，分别计算相应的训练集和测试集分类正确率，最后通过正确率的变化情况确定随机森林所用树的棵数。变化情况如下图所示。

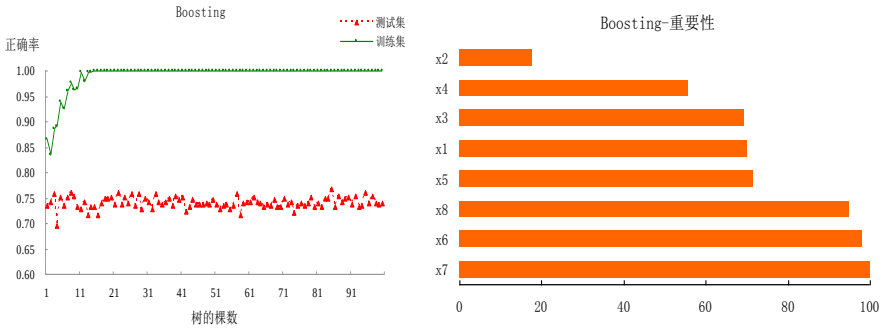


图表 6　随机森林正确率随棵数变化情况（左）变量重要性（右）

与 Bagging 的结果很相似，在随机森林中，训练集的正确率很快上升为 1，而测试集的正确率则稳定在 0.75 左右，并且随着棵数的增加没有出现过拟合的情况。在实际计算中，只要取树的棵数大于 100 棵即可（本文选取 500 棵）。在随机森林中，还需设定每次拆分时随机选取变量个数，一般是不大于自变量个数的平方根，本文每次随机选取的是 2 个。同样，我们也图示了变量相对重要性，含义同 Bagging 法。在 8 个变量中，最重要的仍为 x6，其次为 x7 和 x3，最不重仍是 x2，与 Bagging 结果具有较高的一致性。

（4）Boosting 结果

同样地，Boosting 模型正确率随棵数变化情况变量重要性情况见下图。

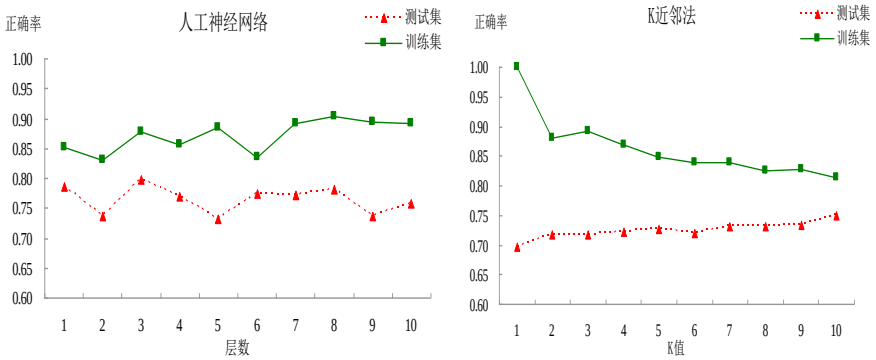


图表 7 Boosting 模型正确率随棵数变化情况（左）变量重要性（右）

由上图可知，在 15 棵数以后，训练集的正确率已达到 100%，而测试集的正确率稳定在 0.73~0.74 左右。正确率要略低于前两种组合方法。考虑到 Booting 的运行速度较慢，本文计算选取的数量为 100 棵，但完全可以保证正确率。从变量重要性来看，在 Boosting 的变量重要性与 Bagging 和随机森林的所有差异，主要体现在最重要的三个变量为 x7、x6 和 x8（其他债务），而 Bagging 和随机森林最重要的三个为 x6、x7 和 x3（现职业从业年限）。

2、人工神经网络模型结果

本文使用的是经典 BP 网络。在 BP 网络中，需要确定隐藏层的层数。层数过少易造成欠拟合，过多则易造成过拟合。本文同样通过训练集和测试集的正确率变化来确定。结果如下图（左）所示。



图表 8 BP 人工神经网络（左）与 K 近邻（右）正确率随层数 K 值变化情况

由上图的测试集正确率可知，当层数为 3 层时，正确率达到最大值。因此，本文 BP 人工神经网络中层数为 3 层。对全样本做 BP 网络时，正确率为 0.82。

3、支持向量机（SVM）模型结果

在 SVM 中，首先需要确定核函数的形式。考虑到线性核函数是径向基核函数的一种特殊情况，而在某些参数下，Sigmoid 核函数与径向基核函数效果相似，因此本文最终选择径向基核函数：$k(\boldsymbol{x},\boldsymbol{y})=\exp(-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|^2)$。在确定模型中的参数$C$，$\varepsilon$及核函数中的$\gamma$时，本文是通过取参数不同的值，在 10 折交叉验证下确定的。最终选取$C=1$，$\varepsilon=0.22$，$\gamma=0.1$，此时对应的全样本正确率为 0.82。

4、K 近邻法模型结果

在 K 近邻法中，核心的参数是 K 值的确定。我们通过改变 K 的取值（从 1 到 10），观察值相应训练集和测试集正确率的变化情况以确定最终的 K 值。由上图（右）可知，随 K 值的增加，测试集正确率也不断增加，K=7 时基本保持稳定，而训练集的正确率在不断下降。为防止过拟合情况，综合考虑后，本文最终取 K=7。在全样本下，该模型的正确率为 0.82。

5、线性判别分析与二次判别分析模型结果

本文的线性判别分析使用的贝叶斯判别分析，对全样本做判别分析时，先验概率为全样本中违约（即 y=1）和非违约（即 y=0）时所占的比重。贝叶斯判别分析与二次判别分析的区别在于前者是假定待分的两类总体协方差矩阵是相同的，而后者假定不相同。对于全样本，贝叶斯判别分析的正确率为 0.81，而二次判别函数的正确率为 0.78。一般而言，当数据的线性趋势较为明显时，贝叶斯判别分析的正确率会高于二次判别分析的。因而，该结果可能意味着本文的数据具有较为明显的线性特征。

6、Logistic 与 Probit 回归结果

我们对全样本做 Logistic 和 Probit 回归，回归结果如下表所示：

表格 3 Logistic 和 Probit 回归结果（全样本）

变量	系数值		标准差		z 值		Pr(>|z|)	
	Logistic	Probit	Logistic	Probit	Logistic	Probit	Logistic	Probit
截距	-1.884	-1.136	0.795	0.455	-2.368	-2.498	0.018*	0.013*
x1	0.035	0.019	0.017	0.010	2.000	1.925	0.046*	0.054
x2	0.043	0.023	0.046	0.027	0.920	0.871	0.358	0.384
x3	-0.257	-0.141	0.033	0.018	-7.771	-7.924	0.000***	0.000***
x4	-0.105	-0.056	0.023	0.013	-4.534	-4.335	0.000***	0.000***
x5	-0.009	-0.004	0.008	0.004	-1.102	-0.923	0.270	0.356
x6	0.068	0.043	0.031	0.017	2.235	2.465	0.025*	0.014*
x7	0.624	0.346	0.113	0.062	5.539	5.601	0.000***	0.000***
x8	0.060	0.021	0.078	0.043	0.771	0.476	0.441	0.634

由上表可知，Logistic 和 Probit 回归的结果具有一致性：变量系数的正负号、变量间相对大小、变量系数显著性等方面基本一致。由变量系数绝对值大小可知，对判定影响最大的三个变量为 x7、x3 和 x4，该结果与基于树类的结果有所差异，主要体现在变量 x4 和 x8 上。在树类结果中，x8 是非常重要的变量。对于全样本，Logistic 的分类正确率为 0.807，Probit 的分类正确率为 0.81，相差无几。

7、单门限 Logistic 回归结果

单门限 Logistic 回归结果如下表所示：

表格 4　门限 Logistic 回归结果（全样本）

变量	系数值		标准差		z 值		Pr(>|z|)	
	区间1	区间2	区间1	区间2	区间1	区间2	区间1	区间2
截距	-1.260	-0.664	1.126	1.695	-1.119	-0.392	0.263	0.695
x1	0.016	0.069	0.024	0.029	0.676	2.420	0.499	0.016**
x2	0.057	0.059	0.060	0.079	0.947	0.750	0.344	0.453
x3	-0.222	-0.299	0.043	0.055	-5.215	-5.399	0.000***	0.000***
x4	-0.102	-0.128	0.033	0.037	-3.099	-3.454	0.002***	0.001***
x5	-0.013	-0.057	0.019	0.034	-0.701	-1.690	0.484	0.091*
x6	0.010	-0.043	0.092	0.075	0.104	-0.575	0.917	0.565
x7	0.751	0.877	0.288	0.219	2.611	4.007	0.009*	0.000***
x8	0.081	0.323	0.267	0.180	0.305	1.794	0.761	0.073
门限变量	x6（家庭负债率）							
门限估计值	12.40							

参考其余模型估计结果，并考虑变量的经济意义，本文最终选取家庭负债率作为门限变量。由上表可知，单门限 Logistic 回归的门限估计值为 12.40，与决策树的结果（12.45）十分类似。从估计结果来看，单门限 Logistic 回归与普通 Logistic 回归的变量系数正负号基本一致，但显著性有一定差别；两个区间系数估计值系数符合、数值大小、显著性均有差别。由变量系数绝对值大小可知，对判定影响最大的三个变量对区间 1 而言为 x7、x3 和 x4；对区间 2 而言为则为 x7、x8 和 x3，与普通 Logistic 回归有差异。对于全样本，门限 Logistic 的分类正确率为 0.82，略高于普通 Logistic 回归。

8、12 个模型全样本运行结果汇总

表格 5 12 个模型全样本运行结果汇总

编号	模型	正确率	最重要的三个变量	重要参数值
1	分类树	0.80	x6,x3,x7	-
2	Bagging	1.00	x6,x3,x7	树的棵数>160
3	随机森林	1.00	x6,x3,x7	树的棵数>100
4	Boosting	1.00	x7,x6,x8	树的棵数>100
5	人工神经网络	0.82	-	隐藏层=3
6	支持向量机	0.82	-	$C=1$， $\varepsilon=0.22$， $\gamma=0.1$
7	K 近邻	0.82	-	K=7
8	线性判别分析	0.81	-	-
9	二次判别分析	0.78	-	-
10	Logistic 回归	0.81	x7,x3,x4	-
11	Probit 回归	0.81	x7,x3,x4	-
12	门限 Logistic 回归	0.82	x7,x3,x4；x7、x8、x3	门限值12.40（家庭负债率）

由上表可知，对于组合模型，正确率为 1；其他模型以门限 Logistic 回归的 0.82 为最高值，但除了二次判别分析略低外，其余正确率也在 0.81 左右。在最重要三个变量方面，模型普遍确定了 x7（信用卡负债）和 x3（现职业从业年限），主要区别在于基于树类的模型还确定了 x6（负债/家庭收入），二元离散选择回归模型确定了 x4（本地生活年限），Boosting 与门限 Logistic 回归还确定了 x8。当然，全样本分析仅用于揭示一些方法的特性。模型间的信用评分能力比较还需通过严谨的方法来最终判定。

五、模型的比较

本部分，我们将通过 10 折交叉验证及预期错误分类损失两种现有文献广泛采用的方法来最终判定 12 个模型在本文情景下的信用评分能力优劣。

（一）10 折交叉验证法

12 个模型的总分类正确率、第一类正确率（好客户被判为好客户的比例）和第二类正确率（坏客户被判为坏客户的比例）平均值如下表所示：

表格 6 12 个模型在 10 折交叉验证下的平均正确率结果摘要

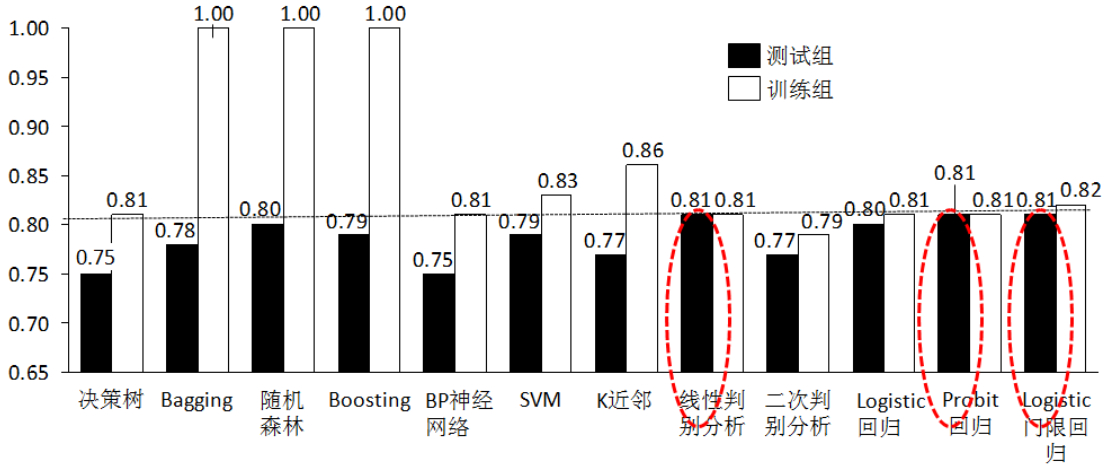
模 型	测试集			训练集		
	正确率	第一类正确率	第二类正确率	正确率	第一类正确率	第二类正确率
决策树	0.75	0.87	0.4	0.81	0.9	0.57
Bagging	0.78	0.89	0.48	1	1	1
随机森林	0.8	0.92	0.46	1	1	1
Boosting	0.79	0.89	0.5	1	1	1
BP 神经网络	0.75	0.86	0.45	0.81	0.9	0.54
SVM	0.79	0.94	0.37	0.83	0.95	0.49
K 近邻	0.77	0.93	0.32	0.86	0.94	0.63
线性判别分析	0.81	0.93	0.47	0.81	0.94	0.46
二次判别分析	0.77	0.92	0.37	0.79	0.92	0.4
Logistic 回归	0.8	0.91	0.49	0.81	0.92	0.5
Probit 回归	0.81	0.92	0.49	0.81	0.92	0.5
Logistic 门限回归	**0.81**	**0.92**	**0.51**	0.82	0.93	0.53

下面我们依据 4 个标准来综合评判在本文背景下信用评分能力排名靠前的几个模型：

1、模型预测能力：训练集与测试集正确率

测试集的高正确率更能反应一个模型的信用评分能力。通过下图，我们可以直观比较各模型在交叉验证下的测试集总分类正确率。

最优前三：门限 Logistic、Probit、线性判别分析（三者并列）



图表 9 12 种方法测试组与训练组正确率

2、模型稳定性 1：训练集与测试集分类总正确率相近程度

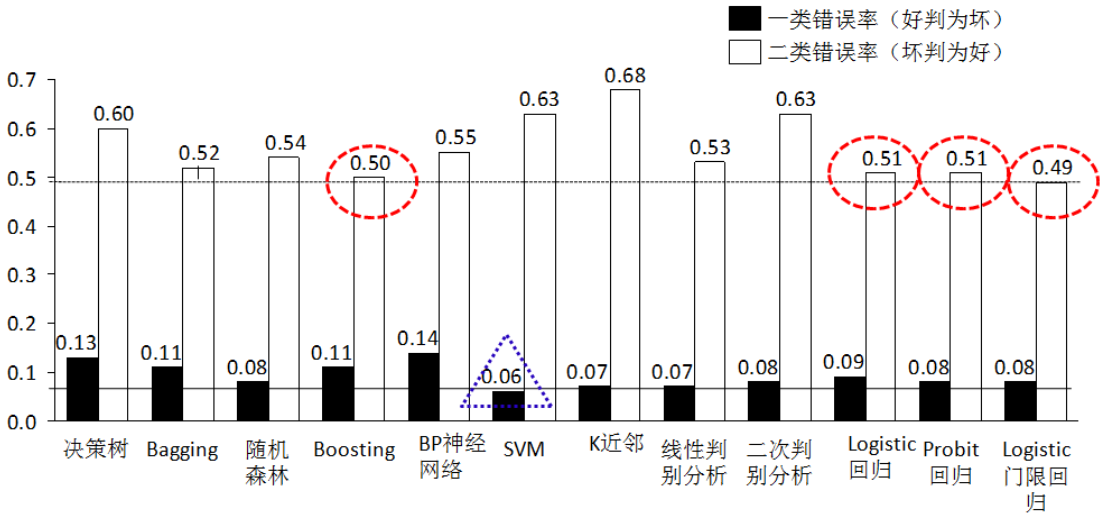
训练集表现与测试集表现的差异可以反映一个模型的稳定性，稳定性差的模型在商业银行实际应用中会受到限制。

最优前三：线性判别分析、probit、Logistic 与门限 Logistic（最后两个并列）

3、模型犯错率：第二类错误率低

从现有结果看，12 类方法在第一类错误上的差异并不大，需从第二类错误上来区分；而且，实践中第二类错误给银行等信贷机构带来的损失成本要显著高于第一类错误（West, 2000)。通过下图，我们可以直观比较各模型在交叉验证下的错误率。

最优前三：门限 Logistic、Boosting、Logistic 和 Probit（最后两个并列）

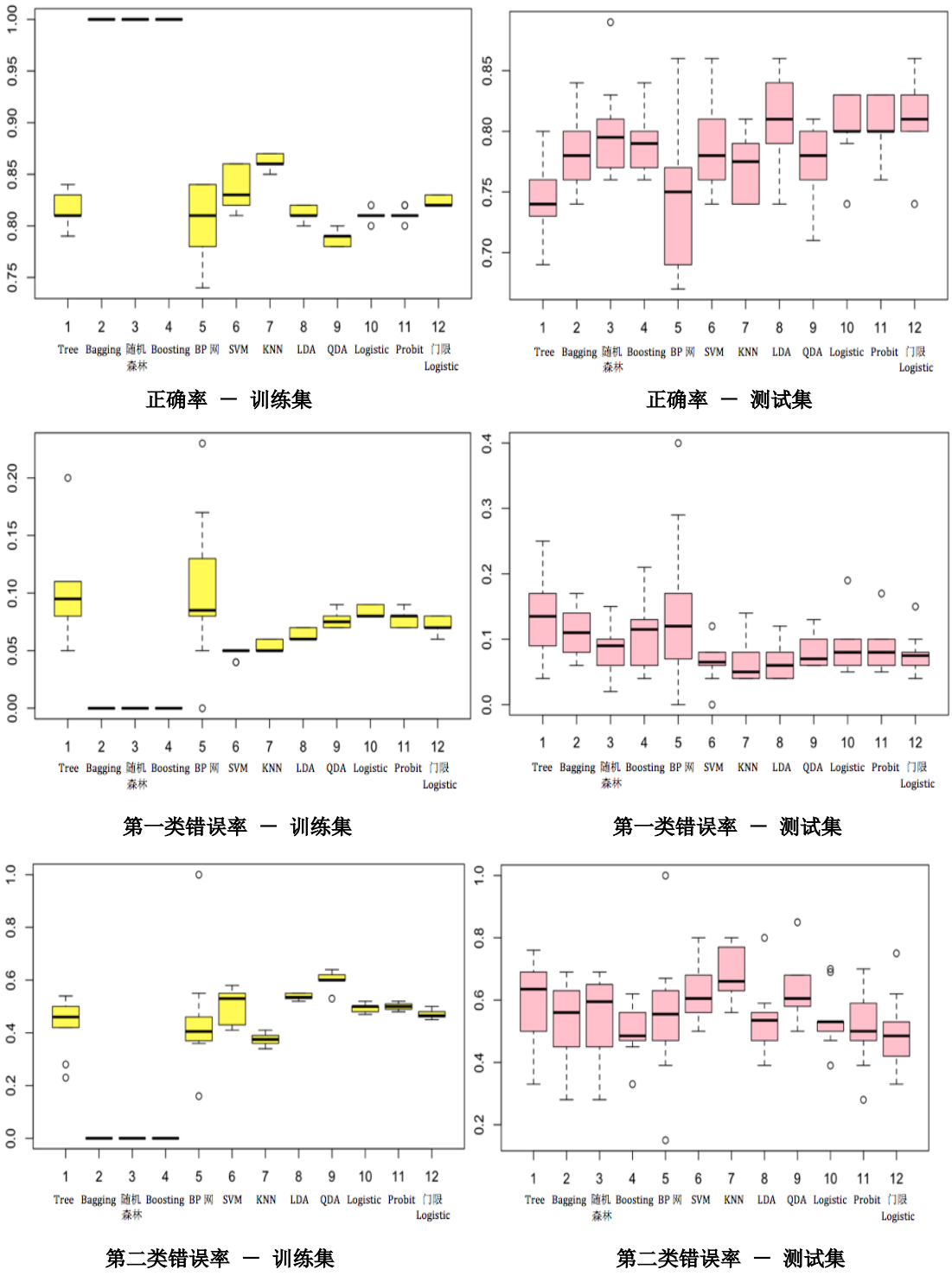


图表 10 12 种方法测试组一类与二类错误率

4、模型稳定性 2：10 折交叉验证结果的离散程度小

10 折交叉验证结果的离散程度小意味着实际应用中模型预测准确率等落入某一较窄范围的概率较大，有利于模型的推广。12 个模型在 10 折交叉验证下的相应结果描述统计箱线图见下图。

最优前三：Logistic、Probit、门限 Logistic。



正确率 — 训练集　　正确率 — 测试集

第一类错误率 — 训练集　　第一类错误率 — 测试集

第二类错误率 — 训练集　　第二类错误率 — 测试集

图表 11 12 个模型在 10 折交叉验证下的相应结果描述统计箱线图

从以上四方面考量，按照出现次数，在本文案例 10 折交叉验证中表现较好的几个模型依次是：门限 Logistic（4）、Logistic（4）、Probit（4）、LDA（2）和 Boosting（1）。

（二）预期错误分类成本法

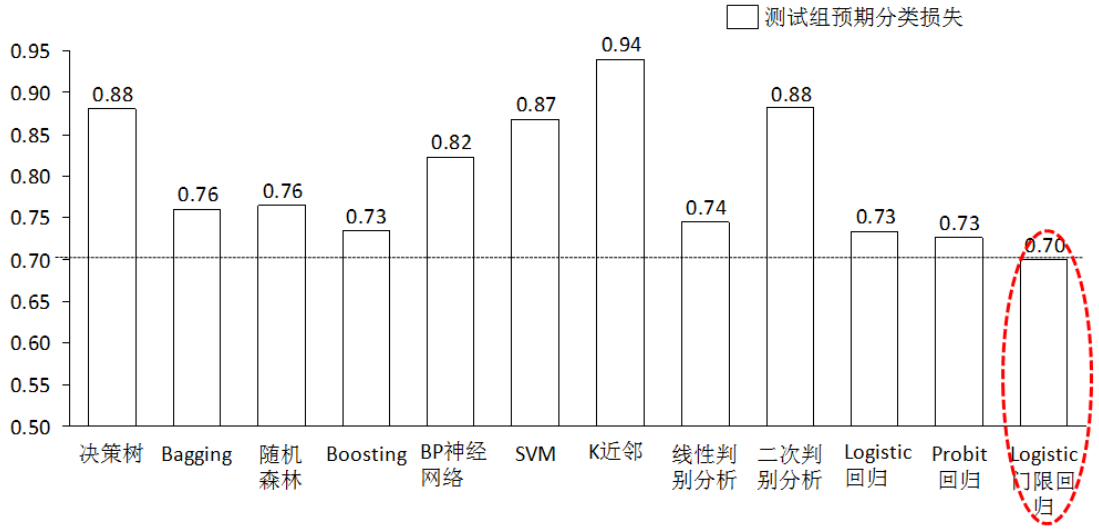
虽然平均正确分类率是评估信用评分模型分类能力的重要依据，但现有文献也普遍提示，对最小预期错误分类成本的考虑也十分必要（Johnson & Wichern, 2002; West, 2000; Lee, 2005; Akkoç, 2012）。公式（4）描述了计算各模型错误分类成本期望值的方程。

$$\cos t = c(2|1) * p(2|1) * \pi_1 + c(1|2) * p(1|2) * \pi_2 \qquad (4)$$

其中 π_1 和 π_2 分别表示客户信用属于“好”和“坏”的先验概率，$p(2|1)$ 和 $p(1|2)$ 分别表示是第一类错误率（一个信用良好的顾客被错误地分到“信用不好的顾客”一类）和第二类错误率（一个信用不好的顾客被错误地分到“信用良好的顾客”一类），$c(2|1)$ 和 $c(1|2)$ 分别是与第一类错误率和第二类错误率相对应的错误分类成本。

为了计算不同评分模型的预期错误分类成本，首先需要求出错误分类概率和错误分类成本的估计值。$p(2|1)$ 和 $p(1|2)$ 最常采用的估计值是，信用良好的顾客被误认为信用不好的比例，和信用不好的顾客被误认为信用良好的比例。由于很难获得有效估计值，因此求错误分类成本的估计值是一项具有挑战性且复杂的任务。然而，在信用评分应用中，大多人认为与第一类错误率和第二类错误率相关的成本是截然不同的。总体来说，与第二类错误率相关的错误分类成本要远高于与第一类错误率相关的错误成本。主笔德国信用数据报告的霍夫曼博士建议，第一类错误和第二类错误相关的分类错误成本是 1 比 5 的关系(West, 2000)。这一分类成本比例关系也被 Lee（2005）等的研究所采纳。本文亦将采用这个相对成本比率来计算本文 12 个信用评分模型的预期错误分类成本。

下图展示了各模型一类错误率、二类错误率及相关的预期错误分类成本。



图表 12　12 种方法测试组与训练组正确率

从上图我们可以直观地看出，在 12 类模型中，本文改进的门限 Logistic 回归，在错误分类成本的期望值方面，表现出了最优的信用评分能力。以决策树为基础通过 boosting 组合方法提高信用评分能力的模型，以及普通 Logistic 回归、Probit 回归也取得了不错的成绩。

因此，从预期错误分类成本来看，排名靠前的几个模型是：门限 Logistic、boosting、Logistic、Probit。

（三）模型比较小结

本小结中，我们尝试把两大类方法的结果进行简单加权来作为最后的模型评定。由于10 折交叉验证法细分了 4 个角度，入选模型得分从 4 分（在各角度模型评优中出现 4 次）到 1 分（出现 1 次），为了同等对待两大类方法结果，我们将预期错误分类成本中评出的模型依据成本数值赋值如下：门限 Logistic（4）、boosting（3）、Logistic（3）、Probit（3）。

因此，综合 10 折交叉验证法以及预期错误分类成本两大类方法，我们最终认为，本文12 个模型的信用评分能力依次是：

门限 Logistic（8）、Logistic（7）、Probit（7）、Boosting（4）和 LDA（2）。

六、基本结论与进一步研究展望

本文的目的在于，从现有方法中广泛选取可能适用于小企业主信用评分领域的多个模型（共计 12 个，包括本文对 Logistic 模型的改进模型-门限 Logistic 模型），并以实际案例来对这些潜在模型的信用评分性能做出全面、客观和严谨的比较，从而为国内商业银行未来做好小企业主信用评分模型的建立和应用提供一定的参考。

在模型比较上，我们遵从现有文献的常用做法，通过 10 折交叉验证和预期分类错误成本的方式，来检验 12 个模型的信用评分能力。分析结果表明，本文改进的门限 Logistic 模型在模型预测能力、稳定性、犯错率及预期错误分类成本等诸多方面表现出了优秀的综合能力；而普通 Logistic、Probit 也表现不俗；而 Boosting 的预期错误分类成本表现仅次于门限 Logistic，显示出较强的信用评分能力，具有较好的推广性；线性判别则在训练集正确率和模型稳定性（训练集与测试集分类总正确率相近程度）上表现出色。

未来的研究将致力于如下几个方面：（1）找寻更多的数据集（特别是国内银行数据）来检验本文结论的适用性；（2）补充更多的可能方法，如二阶段或多阶段组合方法；（3）进一步完善门限 Logistic 回归模型，包括讨论门限变量选取规则，完善相关假设检验等。

最后，需要特别指出的是，小企业主信用评分模型的建立与实施将对银行及政府金融统计工作起到很好的推进工作。一方面，由于模型的建立与实施需要商业银行有大量而翔实的历史客户贷款信息作为基础，商业银行将着手完善其微观层面的金融统计工作；另一方面，商业银行微观统计数据的完善显然也有利于宏观政府金融统计的完善，未来诸如小企业贷款可得性、影响小企业贷款的影响因素等宏观政府金融统计数据有望为政府决策层推进相关金融领域改革提供重要参考。

参考文献

[1] Akkoç S. An empirical comparison of conventional techniques, neural networks and the three stage hybrid Adaptive Neuro Fuzzy Inference System (ANFIS) model for credit scoring analysis: The case of Turkish credit card data. European Journal of Operational Research 222 (2012) 168-178.
[2] Allen N. Berger & W. Scott Frame, 2005. "Small business credit scoring and credit availability," Working Paper 2005-10, Federal Reserve Bank of Atlanta.
[3] Altman E I, Marco G, Varetto. Corporate distress diagnosis: comparison using linear discriminant analysis and neural networks. Journal Banking and Finance,1994,18:505-529.
[4] Arnminger, Enache G D,Bonne T. Analyzing credit risk data:a comparison of logistic discrimination,classification tree analysis and feed-forward networks.Computational Statistics. 1998,12: 293-310.
[5] Baesens B, Van Gestel T,Viaene S,et al. Benchmarking state-of-the-art classification algorithms for credit scoring. Journal of the Operational Research Society, 2003,54: 627-635.
[6] Boyle M,Crook J,Hamilton R,et al. Methods for credit scoring applied to slow payers. Oxford: Clarendon Press, 1992,78-89.
[7] Chatterjee S, Barcun S. A nonparametric approach to credit screening. Journal of the American statistical Association, 1970. 65:150-154
[8] Coffman J Y. The proper role of tree analysis in forecasting the risk behavior of borrowers. Management Decision Systems,1986,3: 47-59.
[9] Cramer J S. Scoring bank loans that may go wrong: a case study. Statistica Neerlandica, 2004,58: 365-380.
[10] Davis R H, Edelman D B,Gammerman A J. Machine-learning algorithms for credit-card applications. Oxford :Oxford University Press, 1992,129-137.
[11] Desai V S,Convay D J, Crook J N. Credit scoring models in the credit union environment using neural networks and genetic algorithms. IMA Journal of Mathematics Applied in Business and Industry, 1997,8: 323-346.
[12] Durand D. Risk Elements in consumer Installment financing. New York: National Bureau of Economic Research. 1941,60-72.
[13] Eisenbeis R A. Pitfalls in the application of discriminant analysis in business, finance, and economics. The Journal of Finance, 1977,32: 875-900.
[14] Eisenbeis R A. Problems in applying discriminant analysis in credit scoring models. Journal of Banking and Finance, 1978,2: 205-209.
[15] Fisher R A.The Use of Multiple Measurement in Taxonomic Problem. Annuals of Eugenic, 1936,7: 179-188.
[16] Frame, S., A. Srinivasan and L. Woosley "The Effect of Credit Scoring on Small Business Lending." Journal of Money, Credit, and Banking. August 2001 (33). Pages 813-25.
[17] Grablowsky B J, Talley W K. Probit and discriminant functions for classifying credit applicants:a comparison. Journal of Economics and Business, 1981,33:254-261.
[18] Hardy W E, Adrian J L. A Linear Programming Alternative to Discriminant Analysis in Credit Scoring. Abribus, 1985,1: 285-292.
[19] Johnson, R. A., & Wichern, D. W. (2002). Applied multivariate statistical analysis (5th ed.). Upper Saddle River, NJ: Prentice-Hall.
[20] Lee T S, Chen I F. A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. Expert Systems with Application 28 (2005) 743-752.
[21]Lee T S,Chiu C C,Chou Y C,et al. Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. Computational Statistics & Data Analysis, 2006. 50(4):1113-1130
[22] Makowski P. Credit scoring branches out. Credit World, 1985,75: 30-37.

[23] Mitchell, Jim, “Automated Scoring Cuts Wait for Loan Decisions,” Dallas Morning News, July 15, 1996, p. 1D.
[24] Morton H. Municipal credit rating modelling by neural networks. Decision Support Systems, 2011,51(1): 108-118
[25] Myers J H, Forgy E W. The development of numerical credit evaluation systems. Journal of the American Statistical Association, 1963, 58: 799-806.
[26] Nath R, Jackson W M, Jones TW. A comparison of the classical and the linear programming approaches to the classification problem in discriminant analysis. J. Statistical Computation and Simulation, 1992, Vol. 41, pp. 73-93.
[27] Odom M,Sharda R. A neural network model for bankruptcy prediction. In: Proceedings of the international joint conference on neural networks. Alamitos,1990,231-245.
[28] Piramuthu S. Financial credit-risk evaluation with neural and neurofuzzy systems. European Journal of Operational Research, 1999,112: 310-321.
[29] West, D. (2000). Neural network credit scoring models. Computers and Operations Research, 27, 1131－1152.
[30] Wiginton J C. A note on the comparison of logit and discriminant models of consumer credit behaviour. Journal of Financial and Quantitative Analysis, 1980,15: 757-770.
[31] Zuckerman, Sam, “Taking Small Business Competition Nationwide,” U.S. Banker, August 1996, p.24.
[32] 石庆焱，靳云汇.多种个人信用评分模型在中国应用的比较研究.统计研究，2004. 6: p. 43-48.
[33] 向晖. 个人信用评分组合模型研究与应用：[湖南大学博士学位论文].长沙：湖南大学，2011.

附件

附件 1：分类树输出结果
附件 2：12 个模型 10 折交叉验证相关结果表
附件 3　R 软件主要程序

附件

附件 1：分类树输出结果

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 700 804.400 0 ( 0.73857 0.26143 )
   2) x6 < 12.45 477 411.700 0 ( 0.84486 0.15514 )
     4) x3 < 5.5 179 217.500 0 ( 0.70391 0.29609 )
       8) x4 < 6.5 116 154.900 0 ( 0.61207 0.38793 )
        16) x6 < 2.35 8   0.000 0 ( 1.00000 0.00000 ) *
        17) x6 > 2.35 108 146.700 0 ( 0.58333 0.41667 ) *
       9) x4 > 6.5 63  47.960 0 ( 0.87302 0.12698 ) *
     5) x3 > 5.5 298 151.900 0 ( 0.92953 0.07047 )
      10) x7 < 0.25 52   0.000 0 ( 1.00000 0.00000 ) *
      11) x7 > 0.25 246 143.500 0 ( 0.91463 0.08537 )
        22) x3 < 14.5 158 116.100 0 ( 0.87975 0.12025 ) *
        23) x3 > 14.5 88  19.090 0 ( 0.97727 0.02273 )
          46) x7 < 5.155 83   0.000 0 ( 1.00000 0.00000 ) *
          47) x7 > 5.155 5   6.730 0 ( 0.60000 0.40000 ) *
   3) x6 > 12.45 223 309.000 0 ( 0.51121 0.48879 )
     6) x3 < 9.5 141 189.500 1 ( 0.39716 0.60284 )
      12) x6 < 26.15 130 177.700 1 ( 0.43077 0.56923 )
        24) x4 < 2.5 29  29.570 1 ( 0.20690 0.79310 ) *
        25) x4 > 2.5 101 140.000 1 ( 0.49505 0.50495 ) *
      13) x6 > 26.15 11   0.000 1 ( 0.00000 1.00000 ) *
     7) x3 > 9.5 82  99.140 0 ( 0.70732 0.29268 )
      14) x7 < 4.905 55  37.910 0 ( 0.89091 0.10909 ) *
      15) x7 > 4.905 27  34.370 1 ( 0.33333 0.66667 )
        30) x8 < 9.225 16  21.930 0 ( 0.56250 0.43750 )
          60) x5 < 73 9   9.535 1 ( 0.22222 0.77778 ) *
          61) x5 > 73 7   0.000 0 ( 1.00000 0.00000 ) *
        31) x8 > 9.225 11   0.000 1 ( 0.00000 1.00000 ) *
```

附件 2：12 个模型 10 折交叉验证相关结果表

附表 1　10 折交叉验证的正确率

第 k 折交叉验证	决策树		Bagging		随机森林		Boosting		BP 神经网络		SVM	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
1	0.81	0.77	1.00	0.74	1.00	0.79	1.00	0.80	0.82	0.83	0.81	0.83
2	0.80	0.69	1.00	0.80	1.00	0.81	1.00	0.77	0.74	0.69	0.86	0.77
3	0.81	0.74	1.00	0.79	1.00	0.83	1.00	0.84	0.84	0.86	0.83	0.86
4	0.83	0.73	1.00	0.74	1.00	0.77	1.00	0.81	0.84	0.73	0.86	0.76
5	0.81	0.74	1.00	0.77	1.00	0.77	1.00	0.77	0.80	0.74	0.82	0.77
6	0.83	0.80	1.00	0.83	1.00	0.76	1.00	0.79	0.78	0.76	0.86	0.76
7	0.81	0.76	1.00	0.77	1.00	0.79	1.00	0.80	0.79	0.67	0.83	0.74
8	0.79	0.73	1.00	0.84	1.00	0.89	1.00	0.79	0.84	0.76	0.82	0.80
9	0.84	0.76	1.00	0.76	1.00	0.80	1.00	0.76	0.83	0.77	0.82	0.81
10	0.81	0.74	1.00	0.79	1.00	0.80	1.00	0.76	0.78	0.69	0.83	0.79

附表 1 续表 1　10 折交叉验证的正确率

第 k 折交叉验证	K 近邻		Logistic 回归		线性判别分析		二次判别分析		Probit 回归		Logistic 门限回归	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
1	0.86	0.74	0.81	0.83	0.80	0.84	0.78	0.80	0.81	0.83	0.82	0.83
2	0.87	0.74	0.81	0.80	0.81	0.79	0.79	0.74	0.81	0.80	0.83	0.80
3	0.85	0.81	0.80	0.81	0.81	0.81	0.78	0.81	0.81	0.80	0.82	0.81
4	0.87	0.79	0.81	0.80	0.82	0.81	0.78	0.80	0.81	0.80	0.82	0.81
5	0.85	0.79	0.81	0.80	0.82	0.79	0.80	0.76	0.82	0.80	0.83	0.81
6	0.86	0.76	0.81	0.83	0.81	0.83	0.79	0.77	0.80	0.83	0.82	0.86
7	0.86	0.74	0.82	0.74	0.82	0.74	0.79	0.71	0.82	0.76	0.83	0.74
8	0.86	0.80	0.81	0.83	0.81	0.86	0.78	0.80	0.81	0.83	0.82	0.84
9	0.86	0.79	0.81	0.80	0.81	0.84	0.79	0.79	0.81	0.81	0.82	0.83
10	0.87	0.76	0.81	0.79	0.82	0.80	0.79	0.76	0.81	0.80	0.83	0.80

附表 2　10 折交叉验证的第一类错误率

第 k 折交叉验证	决策树		Bagging		随机森林		Boosting		BP 神经网络		SVM	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
1	0.11	0.12	0.00	0.16	0.00	0.11	0.00	0.11	0.09	0.07	0.05	0.07
2	0.11	0.17	0.00	0.08	0.00	0.06	0.00	0.13	0.00	0.00	0.05	0.06
3	0.07	0.09	0.00	0.09	0.00	0.02	0.00	0.06	0.08	0.04	0.05	0.00
4	0.05	0.12	0.00	0.12	0.00	0.10	0.00	0.04	0.08	0.14	0.05	0.08
5	0.11	0.17	0.00	0.10	0.00	0.10	0.00	0.13	0.13	0.17	0.05	0.08
6	0.08	0.04	0.00	0.06	0.00	0.08	0.00	0.12	0.10	0.12	0.04	0.06
7	0.08	0.06	0.00	0.06	0.00	0.04	0.00	0.06	0.23	0.40	0.05	0.04
8	0.20	0.25	0.00	0.12	0.00	0.06	0.00	0.10	0.05	0.10	0.05	0.08
9	0.11	0.16	0.00	0.14	0.00	0.10	0.00	0.16	0.08	0.12	0.05	0.06
10	0.08	0.15	0.00	0.17	0.00	0.15	0.00	0.21	0.17	0.29	0.04	0.12

附表 2 续表 1　10 折交叉验证的第一类错误率

第 k 折交叉验证	K 近邻		Logistic 回归		线性判别分析		二次判别分析		Probit 回归		Logistic 门限回归	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
1	0.05	0.14	0.09	0.05	0.07	0.07	0.07	0.11	0.08	0.05	0.08	0.07
2	0.05	0.08	0.08	0.06	0.07	0.06	0.08	0.06	0.08	0.06	0.07	0.06
3	0.06	0.04	0.08	0.08	0.06	0.06	0.08	0.06	0.07	0.08	0.08	0.08
4	0.05	0.04	0.09	0.08	0.06	0.08	0.08	0.06	0.09	0.08	0.08	0.08
5	0.06	0.08	0.08	0.10	0.06	0.10	0.09	0.10	0.07	0.10	0.06	0.08
6	0.06	0.04	0.08	0.06	0.07	0.04	0.08	0.06	0.08	0.06	0.07	0.04
7	0.05	0.04	0.08	0.08	0.06	0.04	0.07	0.06	0.07	0.06	0.07	0.06
8	0.05	0.04	0.08	0.10	0.06	0.06	0.07	0.10	0.08	0.10	0.07	0.10
9	0.05	0.06	0.09	0.08	0.07	0.04	0.07	0.08	0.08	0.08	0.08	0.06
10	0.06	0.13	0.08	0.19	0.06	0.12	0.07	0.13	0.08	0.17	0.06	0.15

附表 3　10 折交叉验证的第二类错误率

第 k 折交叉验证	决策树		Bagging		随机森林		Boosting		BP 神经网络		SVM	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
1	0.42	0.69	0.00	0.69	0.00	0.69	0.00	0.62	0.41	0.62	0.58	0.62
2	0.48	0.64	0.00	0.45	0.00	0.45	0.00	0.45	1.00	1.00	0.41	0.59
3	0.54	0.76	0.00	0.59	0.00	0.65	0.00	0.47	0.40	0.47	0.51	0.59
4	0.52	0.68	0.00	0.63	0.00	0.58	0.00	0.58	0.37	0.63	0.43	0.68
5	0.43	0.50	0.00	0.61	0.00	0.61	0.00	0.50	0.39	0.50	0.54	0.67
6	0.44	0.63	0.00	0.47	0.00	0.68	0.00	0.47	0.55	0.58	0.41	0.74
7	0.48	0.70	0.00	0.65	0.00	0.65	0.00	0.55	0.16	0.15	0.52	0.80
8	0.23	0.33	0.00	0.28	0.00	0.28	0.00	0.56	0.46	0.67	0.55	0.56
9	0.28	0.47	0.00	0.53	0.00	0.47	0.00	0.47	0.46	0.53	0.57	0.53
10	0.50	0.56	0.00	0.33	0.00	0.33	0.00	0.33	0.36	0.39	0.54	0.50

附表 3 续表 1　10 折交叉验证的第二类错误率

第 k 折交叉验证	K 近邻		Logistic 回归		线性判别分析		二次判别分析		Probit 回归		Logistic 门限回归	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
1	0.36	0.77	0.48	0.69	0.54	0.54	0.62	0.62	0.49	0.69	0.46	0.62
2	0.37	0.64	0.50	0.50	0.54	0.55	0.60	0.68	0.50	0.50	0.46	0.50
3	0.38	0.65	0.51	0.53	0.53	0.59	0.62	0.59	0.50	0.59	0.48	0.53
4	0.34	0.68	0.48	0.53	0.53	0.47	0.60	0.58	0.49	0.53	0.47	0.47
5	0.38	0.61	0.50	0.50	0.52	0.56	0.53	0.67	0.50	0.50	0.48	0.50
6	0.36	0.79	0.52	0.47	0.55	0.53	0.60	0.68	0.52	0.47	0.49	0.42
7	0.40	0.80	0.47	0.70	0.52	0.80	0.60	0.85	0.48	0.70	0.45	0.75
8	0.41	0.67	0.50	0.39	0.55	0.39	0.64	0.50	0.52	0.39	0.50	0.33
9	0.39	0.63	0.50	0.53	0.55	0.47	0.61	0.58	0.51	0.47	0.46	0.47
10	0.35	0.56	0.50	0.28	0.53	0.44	0.60	0.56	0.51	0.28	0.46	0.33

附表 4　10 折交叉验证下正确率的描述统计

统计量	决策树		Bagging		随机森林		Boosting		BP 神经网络		SVM	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
平均值	0.81	0.75	1.00	0.78	1.00	0.80	1.00	0.79	0.81	0.75	0.83	0.79
标准差	0.02	0.03	0.00	0.03	0.00	0.04	0.00	0.03	0.03	0.06	0.02	0.04
最小值	0.79	0.69	1.00	0.74	1.00	0.76	1.00	0.76	0.74	0.67	0.81	0.74
下四分位数（25%）	0.81	0.73	1.00	0.76	1.00	0.78	1.00	0.77	0.78	0.70	0.82	0.76
中位数（50%）	0.81	0.74	1.00	0.78	1.00	0.79	1.00	0.79	0.81	0.75	0.83	0.78
上四分位数（75%）	0.82	0.76	1.00	0.80	1.00	0.81	1.00	0.80	0.83	0.77	0.85	0.81
最大值	0.84	0.80	1.00	0.84	1.00	0.89	1.00	0.84	0.84	0.86	0.86	0.86

附表 4 续表 1　　10 折交叉验证下正确率的描述统计

统计量	K 近邻		线性判别分析		二次判别分析		Logistic 回归		Probit 回归		Logistic 门限回归	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
平均值	0.86	0.77	0.81	0.81	0.79	0.77	0.81	0.80	0.81	0.81	0.82	0.81
标准差	0.01	0.03	0.01	0.03	0.01	0.03	0.00	0.03	0.00	0.02	0.01	0.03
最小值	0.85	0.74	0.80	0.74	0.78	0.71	0.80	0.74	0.80	0.76	0.82	0.74
下四分位数（25%）	0.86	0.75	0.81	0.79	0.78	0.76	0.81	0.80	0.81	0.80	0.82	0.80
中位数（50%）	0.86	0.77	0.81	0.81	0.79	0.78	0.81	0.80	0.81	0.80	0.82	0.81
上四分位数（75%）	0.87	0.79	0.82	0.84	0.79	0.80	0.81	0.83	0.81	0.83	0.83	0.83
最大值	0.87	0.81	0.82	0.86	0.80	0.81	0.82	0.83	0.82	0.83	0.83	0.86

附表 5　10 折交叉验证下第一类错误率的描述统计

统计量	决策树		Bagging		随机森林		Boosting		BP 神经网络		SVM	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
平均值	0.10	0.13	0.00	0.11	0.00	0.08	0.00	0.11	0.10	0.14	0.05	0.06
标准差	0.04	0.06	0.00	0.04	0.00	0.04	0.00	0.05	0.06	0.12	0.00	0.03
最小值	0.05	0.04	0.00	0.06	0.00	0.02	0.00	0.04	0.00	0.00	0.04	0.00
下四分位数（25%）	0.08	0.10	0.00	0.09	0.00	0.06	0.00	0.07	0.08	0.08	0.05	0.06
中位数（50%）	0.09	0.14	0.00	0.11	0.00	0.09	0.00	0.11	0.09	0.12	0.05	0.07
上四分位数（75%）	0.11	0.16	0.00	0.13	0.00	0.10	0.00	0.13	0.13	0.16	0.05	0.08
最大值	0.20	0.25	0.00	0.17	0.00	0.15	0.00	0.21	0.23	0.40	0.05	0.12

附表 5 续表 1　10 折交叉验证下第一类错误率的描述统计

统计量	K 近邻		线性判别分析		二次判别分析		Logistic 回归		Probit 回归		Logistic 门限回归	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
平均值	0.06	0.07	0.06	0.07	0.08	0.08	0.08	0.09	0.08	0.08	0.07	0.08
标准差	0.01	0.04	0.00	0.03	0.00	0.03	0.00	0.04	0.00	0.03	0.01	0.03
最小值	0.05	0.04	0.06	0.04	0.07	0.06	0.08	0.05	0.07	0.05	0.06	0.04
下四分位数（25%）	0.05	0.04	0.06	0.04	0.07	0.06	0.08	0.07	0.08	0.06	0.07	0.06
中位数（50%）	0.05	0.05	0.06	0.06	0.07	0.07	0.08	0.08	0.08	0.08	0.07	0.08
上四分位数（75%）	0.06	0.08	0.07	0.08	0.08	0.10	0.09	0.09	0.08	0.09	0.08	0.08
最大值	0.06	0.14	0.07	0.12	0.09	0.13	0.09	0.19	0.09	0.17	0.08	0.15

附表 6　10 折交叉验证下第二类错误率的描述统计

统计量	决策树		Bagging		随机森林		Boosting		BP 神经网络		SVM	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
平均值	0.43	0.60	0.00	0.52	0.00	0.54	0.00	0.50	0.46	0.55	0.51	0.63
标准差	0.10	0.13	0.00	0.14	0.00	0.15	0.00	0.08	0.22	0.22	0.06	0.09
最小值	0.23	0.33	0.00	0.28	0.00	0.28	0.00	0.33	0.16	0.15	0.41	0.50
下四分位数（25%）	0.42	0.51	0.00	0.46	0.00	0.46	0.00	0.47	0.38	0.48	0.45	0.56
中位数（50%）	0.46	0.63	0.00	0.56	0.00	0.60	0.00	0.49	0.40	0.55	0.53	0.60
上四分位数（75%）	0.50	0.69	0.00	0.63	0.00	0.65	0.00	0.55	0.46	0.63	0.55	0.68
最大值	0.54	0.76	0.00	0.69	0.00	0.69	0.00	0.62	1.00	1.00	0.58	0.80

附表 6 续表 1　10 折交叉验证下第二类错误率的描述统计

统计量	K 近邻		线性判别分析		二次判别分析		Logistic 回归		Probit 回归		Logistic 门限回归	
	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集	训练集	测试集
平均值	0.37	0.68	0.54	0.53	0.60	0.63	0.50	0.51	0.50	0.51	0.47	0.49
标准差	0.02	0.08	0.01	0.11	0.03	0.10	0.01	0.12	0.01	0.13	0.02	0.13
最小值	0.34	0.56	0.52	0.39	0.53	0.50	0.47	0.28	0.48	0.28	0.45	0.33
下四分位数（25%）	0.36	0.63	0.53	0.47	0.60	0.58	0.49	0.48	0.49	0.47	0.46	0.43
中位数（50%）	0.37	0.66	0.54	0.53	0.60	0.60	0.50	0.51	0.50	0.50	0.47	0.49
上四分位数（75%）	0.39	0.75	0.55	0.55	0.62	0.68	0.50	0.53	0.51	0.57	0.48	0.52
最大值	0.41	0.80	0.55	0.80	0.64	0.85	0.52	0.70	0.52	0.70	0.50	0.75

附件 3　R 软件主要程序

```
#计算全样本下的模型结果

#普通的决策树

  rtree2<-tree(y~.,data1) #训练树
  ryy2<-predict(rtree2,newdata=data1,type="class") #计算预测值
  rtest2<-data1[,"y"] #实际值
  rbiaoge2<-table(ryy2,rtest2) #建立预测值与实际值的列联表
  rcorrate2<-sum(diag(rbiaoge2))/sum(rbiaoge2) #对脚线元素之和为
正确分类的个数



#Bagging

  rtree3<-randomForest(y~.,data1,mtry=8) #mtry 的值为所有自变量个
数
  ryy2<-predict(rtree3,newdata=data1,type="class")
  rtest2<-data1[,"y"]
  rbiaoge2<-table(ryy2,rtest2)
  rcorrate2<-sum(diag(rbiaoge2))/sum(rbiaoge2)



#随机森林
  rtree4<-randomForest(y~.,data1)
  ryy2<-predict(rtree4,newdata=data1,type="class")
  rtest2<-data1[,"y"]
  rbiaoge2<-table(ryy2,rtest2)
  rcorrate2<-sum(diag(rbiaoge2))/sum(rbiaoge2)


#K 近邻

  ryy2<-knn(data1,data1,data1[,"y"],k=3) #knn(训练集，测试集，训练
集类标号，近邻数)
  rtest2<-data1[,"y"] #实际值
  rbiaoge2<-table(ryy2,rtest2) #建立预测值与实际值的列联表
  rcorrate2<-sum(diag(rbiaoge2))/sum(rbiaoge2) #对脚线元素之和为
正确分类的个数
```

```
#SVM

svmtrain<-tune(svm,y~.,data=data1,kernel="radial",ranges=list(cost=c(0.1,1,10,100,1000),gamma=c(0.1,0.5,1,2,3,4,5,7,7,8,9,10))) #通过交叉验证选取合适的 cost 和 gamma 值
    ryy2<-predict(svmtrain$best.model,newdata=data1) #计算预测值
    rtest2<-data1[,"y"] #实际值
    rbiaoge2<-table(ryy2,rtest2) #建立预测值与实际值的列联表
    rcorrate2<-sum(diag(rbiaoge2))/sum(rbiaoge2) #对脚线元素之和为正确分类的个数



#Logistic 回归

    rglmtrain<-glm(y~.,data1,family=binomial) #要设定 familiy 的值
    rglmpred2<-predict(rglmtrain,newdata=data1,type="response") #求出 y=1 的概率
    ryy2<-ifelse(rglmpred2>.5,1,0) #概率值大于 0.5 时认为属于第 1 类
    rtest2<-data1[,"y"]
    rbiaoge2<-table(ryy2,rtest2)
    rcorrate2<-sum(diag(rbiaoge2))/sum(rbiaoge2)



#线性判别分析 lda

    rldatrain<-lda(y~.,data1) #训练样本
    rldapred2<-predict(rldatrain,newdata=data1) #计算预测值
    rlyy2<-rldapred2$class
    rltest2<-data1[,"y"] #实际值
    rlbiaoge2<-table(rlyy2,rltest2) #建立预测值与实际值的列联表
    rlcorrate2<-sum(diag(rlbiaoge2))/sum(rlbiaoge2) #对脚线元素之和为正确分类的个数



#线性判别分析 qda

    qdatrain<-qda(y~.,data1) #训练样本
    qdapred2<-predict(qdatrain,newdata=data1) #计算预测值
    rqyy2<-qdapred2$class
    rqtest2<-data1[,"y"] #实际值
    rqbiaoge2<-table(rqyy2,rqtest2) #建立预测值与实际值的列联表
```

```
    rqcorrate2<-sum(diag(rqbiaoge2))/sum(rqbiaoge2) #对脚线元素之和为正确分类的个数


#Probit 回归

    rpglmtrain<-glm(y~.,data1,family=binomial(link="probit")) #要设定 familiy 的值
    rpglmpred2<-predict(rpglmtrain,newdata=data1,type="response") #求出 y=1 的概率
    rpyy2<-ifelse(rpglmpred2>.5,1,0) #概率值大于 0.5 时认为属于第 1 类
    rptest2<-data1[,"y"]
    rpbiaoge2<-table(rpyy2,rptest2) #建立预测值与实际值的列联表
    rpcorrate2<-sum(diag(rpbiaoge2))/sum(rpbiaoge2) #对脚线元素之和为正确分类的个数


#Boosting

    rboostingtrain<-boosting(y~.,data1) #训练树
    rboostingpred2<-predict(rboostingtrain,newdata=data1) #计算预测值
    ryy2<-rboostingpred2$class
    rtest2<-data1[,"y"] #实际值
    rbiaoge2<-table(ryy2,rtest2) #建立预测值与实际值的列联表
    rcorrate2<-sum(diag(rbiaoge2))/sum(rbiaoge2) #对脚线元素之和为正确分类的个数

#SVM

rsvmtrain<-tune(svm,y~.,data=data1,kernel="radial",ranges=list(cost=c(0.1,1,10,100,1000),gamma=c(0.1,0.5,1,2,3,4,5,7,7,8,9,10))) #通过交叉验证选取合适的 cost 和 gamma 值
    ryy2<-predict(rsvmtrain$best.model,newdata=data1) #计算预测值
    rtest2<-data1[,"y"] #实际值
    rbiaoge2<-table(ryy2,rtest2) #建立预测值与实际值的列联表
    rcorrate2<-sum(diag(rbiaoge2))/sum(rbiaoge2) #对脚线元素之和为正确分类的个数


#BP 人工神经网络
```

```
      rbb<-class.ind(data1$y)#生成输出值标号矩阵

rnnettrain<-nnet(data1[,-1],rbb,size=3,rang=0.1,decay=5e-4,maxit=200)
#训练网络 nnet(X,Y,...)
      rnnetpred2<-predict(rnnettrain,data1[,-1]) #计算预测值
      ryy2<-max.col(rnnetpred2)
      rtest2<-max.col(rbb) #实际值
      rbiaoge2<-table(ryy2,rtest2) #建立预测值与实际值的列联表
      rcorrate2<-sum(diag(rbiaoge2))/sum(rbiaoge2) #对脚线元素之和为
正确分类的个数

    #测试树的过拟合情况
    nshushu=100
    for(i in 1:nshushu){
      nboostingtrain<-boosting(y~.,data1[-nceshi,],mfinal=i) #训练树
      nboostingpred<-predict(nboostingtrain,newdata=data1[nceshi,]) #
计算预测值
      nyy<-nboostingpred$class
      ntest<-data1[nceshi,"y"] #实际值
      nbiaoge[[1]]<-table(nyy,ntest) #建立预测值与实际值的列联表
      ncorrate[[1]]<-sum(diag(nbiaoge[[1]]))/sum(nbiaoge[[1]]) #对脚
线元素之和为正确分类的个数

nresult[[1]]<-c(nresult[[1]],ncorrate[[1]]<-sum(diag(nbiaoge[[1]]))/s
um(nbiaoge[[1]])) #在交叉验证下求出各折的正确率
      nboostingpred2<-predict(nboostingtrain,newdata=data1[-nceshi,])
#计算预测值
      nyy2<-nboostingpred2$class
      ntest2<-data1[-nceshi,"y"] #实际值
      nbiaoge2[[1]]<-table(nyy2,ntest2) #建立预测值与实际值的列联表
      ncorrate2[[1]]<-sum(diag(nbiaoge2[[1]]))/sum(nbiaoge2[[1]]) #对
脚线元素之和为正确分类的个数

nresult2[[1]]<-c(nresult2[[1]],ncorrate2[[1]]<-sum(diag(nbiaoge2[[1]]
))/sum(nbiaoge2[[1]])) #在交叉验证下求出各折的正确率
    }

    #Bagging
    for(i in 1:nshu){
      ntree3<-randomForest(y~.,data1[-nceshi,],mtry=8,ntree=i) #mtry
的值为所有自变量个数
      nyy<-predict(ntree3,newdata=data1[nceshi,],type="class")
      ntest<-data1[nceshi,"y"]
      nbiaoge[[2]]<-table(nyy,ntest)
```

```
      ncorrate[[2]]<-sum(diag(nbiaoge[[2]]))/sum(nbiaoge[[2]])

nresult[[2]]<-c(nresult[[2]],ncorrate[[2]]<-sum(diag(nbiaoge[[2]]))/s
um(nbiaoge[[2]]))
      nyy2<-predict(ntree3,newdata=data1[-nceshi,],type="class")
      ntest2<-data1[-nceshi,"y"]
      nbiaoge2[[2]]<-table(nyy2,ntest2)
      ncorrate2[[2]]<-sum(diag(nbiaoge2[[2]]))/sum(nbiaoge2[[2]])

nresult2[[2]]<-c(nresult2[[2]],ncorrate2[[2]]<-sum(diag(nbiaoge2[[2]]
))/sum(nbiaoge2[[2]]))
    }


    #随机森林
    for(i in 1:nshu){
      ntree4<-randomForest(y~.,data1[-nceshi,],ntree=i)
      nyy<-predict(ntree4,newdata=data1[nceshi,],type="class")
      ntest<-data1[nceshi,"y"]
      nbiaoge[[3]]<-table(nyy,ntest)
      ncorrate[[3]]<-sum(diag(nbiaoge[[3]]))/sum(nbiaoge[[3]])

nresult[[3]]<-c(nresult[[3]],ncorrate[[3]]<-sum(diag(nbiaoge[[3]]))/s
um(nbiaoge[[3]]))
      nyy2<-predict(ntree4,newdata=data1[-nceshi,],type="class")
      ntest2<-data1[-nceshi,"y"]
      nbiaoge2[[3]]<-table(nyy2,ntest2)
      ncorrate2[[3]]<-sum(diag(nbiaoge2[[3]]))/sum(nbiaoge2[[3]])

nresult2[[3]]<-c(nresult2[[3]],ncorrate2[[3]]<-sum(diag(nbiaoge2[[3]]
))/sum(nbiaoge2[[3]]))
    }


    #BP 人工神经网络
    nceng=10
    for(i in 1:nceng){
      nbb<-class.ind(data1$y)#生成输出值标号矩阵

nnnettrain<-nnet(data1[-nceshi,-1],nbb[-nceshi,],size=i,rang=0.1,deca
y=5e-4,maxit=200) #训练网络 nnet(X,Y,...)
      nnnetpred<-predict(nnnettrain,data1[nceshi,-1]) #计算预测值
      nyy<-max.col(nnnetpred)
      ntest<-max.col(nbb[nceshi,])
```

```
      nbiaoge[[4]]<- table(nyy,ntest) #建立预测值与实际值的列联表
      ncorrate[[4]]<-sum(diag(nbiaoge[[4]]))/sum(nbiaoge[[4]]) #对脚
线元素之和为正确分类的个数

nresult[[4]]<-c(nresult[[4]],ncorrate[[4]]<-sum(diag(nbiaoge[[4]]))/s
um(nbiaoge[[4]])) #在交叉验证下求出各折的正确率
      nnnetpred2<-predict(nnnettrain,data1[-nceshi,-1]) #计算预测值
      nyy2<-max.col(nnnetpred2)
      ntest2<-max.col(nbb[-nceshi,]) #实际值
      nbiaoge2[[4]]<-table(nyy2,ntest2) #建立预测值与实际值的列联表
      ncorrate2[[4]]<-sum(diag(nbiaoge2[[4]]))/sum(nbiaoge2[[4]]) #对
脚线元素之和为正确分类的个数

nresult2[[4]]<-c(nresult2[[4]],ncorrate2[[4]]<-sum(diag(nbiaoge2[[4]]
))/sum(nbiaoge2[[4]])) #在交叉验证下求出各折的正确率
    }

    #K 近邻
    jinlin<-10
    for(i in 1:jinlin){
      nyy<-knn(data1[-nceshi,],data1[nceshi,],data1[-nceshi,"y"],k=i)
#knn(训练集，测试集，训练集类标号，近邻数)
      ntest<-data1[nceshi,"y"] #实际值
      nbiaoge[[5]]<-table(nyy,ntest) #建立预测值与实际值的列联表
      ncorrate4<-sum(diag(nbiaoge[[5]]))/sum(nbiaoge[[5]]) #对脚线元
素之和为正确分类的个数

nresult[[5]]<-c(nresult[[5]],ncorrate[[5]]<-sum(diag(nbiaoge[[5]]))/s
um(nbiaoge[[5]])) #在交叉验证下求出各折的正确率

nyy2<-knn(data1[-nceshi,],data1[-nceshi,],data1[-nceshi,"y"],k=i)
#knn(训练集，测试集，训练集类标号，近邻数)
      ntest2<-data1[-nceshi,"y"] #实际值
      nbiaoge2[[5]]<-table(nyy2,ntest2) #建立预测值与实际值的列联表
      ncorrate2[[5]]<-sum(diag(nbiaoge2[[5]]))/sum(nbiaoge2[[4]]) #对
脚线元素之和为正确分类的个数

nresult2[[5]]<-c(nresult2[[5]],ncorrate2[[5]]<-sum(diag(nbiaoge2[[5]]
))/sum(nbiaoge2[[5]])) #在交叉验证下求出各折的正确率
    }


    #交叉验证
    #普通的决策树
```

```
    for(i in 1:k){
     tree2<-tree(y~.,data1[-dat[,i],]) #训练树
      yy<-predict(tree2,newdata=data1[dat[,i],],type="class") #计算预测值
      test<-data1[dat[,i],"y"] #实际值
      biaoge[[1]]<-table(yy,test) #建立预测值与实际值的列联表
      typei[[1]]<-c(typei[[1]],biaoge[[1]][2,1]/sum(biaoge[[1]][,1])) #计算预测值的第一类错误率

typeii[[1]]<-c(typeii[[1]],biaoge[[1]][1,2]/sum(biaoge[[1]][,2])) #计算预测值的第二类错误率
      corrate[[1]]<-sum(diag(biaoge[[1]]))/sum(biaoge[[1]]) #对脚线元素之和为正确分类的个数

result[[1]]<-c(result[[1]],corrate[[1]]<-sum(diag(biaoge[[1]]))/sum(biaoge[[1]])) #在交叉验证下求出各折的正确率
      yy2<-predict(tree2,newdata=data1[-dat[,i],],type="class") #计算预测值
      test2<-data1[-dat[,i],"y"] #实际值
      biaoge2[[1]]<-table(yy2,test2) #建立预测值与实际值的列联表

typei2[[1]]<-c(typei2[[1]],biaoge2[[1]][2,1]/sum(biaoge2[[1]][,1])) #计算训练值的第一类错误率

typeii2[[1]]<-c(typeii2[[1]],biaoge2[[1]][1,2]/sum(biaoge2[[1]][,2])) #计算训练值的第二类错误率
      corrate2[[1]]<-sum(diag(biaoge2[[1]]))/sum(biaoge2[[1]]) #对脚线元素之和为正确分类的个数

result2[[1]]<-c(result2[[1]],corrate2[[1]]<-sum(diag(biaoge2[[1]]))/sum(biaoge2[[1]])) #在交叉验证下求出各折的正确率
    }

    #Bagging
    for(i in 1:k){
      tree3<-randomForest(y~.,data1[-dat[,i],],mtry=8) #mtry 的值为所有自变量个数
      yy<-predict(tree3,newdata=data1[dat[,i],],type="class")
      test<-data1[dat[,i],"y"]
      biaoge[[2]]<-table(yy,test)
      typei[[2]]<-c(typei[[2]],biaoge[[2]][2,1]/sum(biaoge[[2]][,1]))

typeii[[2]]<-c(typeii[[2]],biaoge[[2]][1,2]/sum(biaoge[[2]][,2]))
      corrate[[2]]<-sum(diag(biaoge[[2]]))/sum(biaoge[[2]])
```

```
result[[2]]<-c(result[[2]],corrate[[2]]<-sum(diag(biaoge[[2]]))/sum(b
iaoge[[2]]))
      yy2<-predict(tree3,newdata=data1[-dat[,i],],type="class")
      test2<-data1[-dat[,i],"y"]
      biaoge2[[2]]<-table(yy2,test2)

typei2[[2]]<-c(typei2[[2]],biaoge2[[2]][2,1]/sum(biaoge2[[2]][,1]))

typeii2[[2]]<-c(typeii2[[2]],biaoge2[[2]][1,2]/sum(biaoge2[[2]][,2]))
      corrate2[[2]]<-sum(diag(biaoge2[[2]]))/sum(biaoge2[[2]])

result2[[2]]<-c(result2[[2]],corrate2[[2]]<-sum(diag(biaoge2[[2]]))/s
um(biaoge2[[2]]))
    }


    #随机森林
    for(i in 1:k){
      tree4<-randomForest(y~.,data1[-dat[,i],])
      yy<-predict(tree4,newdata=data1[dat[,i],],type="class")
      test<-data1[dat[,i],"y"]
      biaoge[[3]]<-table(yy,test)
      typei[[3]]<-c(typei[[3]],biaoge[[3]][2,1]/sum(biaoge[[3]][,1]))

typeii[[3]]<-c(typeii[[3]],biaoge[[3]][1,2]/sum(biaoge[[3]][,2]))
      corrate[[3]]<-sum(diag(biaoge[[3]]))/sum(biaoge[[3]])

result[[3]]<-c(result[[3]],corrate[[3]]<-sum(diag(biaoge[[3]]))/sum(b
iaoge[[3]]))
      yy2<-predict(tree4,newdata=data1[-dat[,i],],type="class")
      test2<-data1[-dat[,i],"y"]
      biaoge2[[3]]<-table(yy2,test2)

typei2[[3]]<-c(typei2[[3]],biaoge2[[3]][2,1]/sum(biaoge2[[3]][,1]))

typeii2[[3]]<-c(typeii2[[3]],biaoge2[[3]][1,2]/sum(biaoge2[[3]][,2]))
      corrate2[[3]]<-sum(diag(biaoge2[[3]]))/sum(biaoge2[[3]])

result2[[3]]<-c(result2[[3]],corrate2[[3]]<-sum(diag(biaoge2[[3]]))/s
um(biaoge2[[3]]))
    }

    #K 近邻
```

```
    for(i in 1:k){

yy<-knn(data1[-dat[,i],],data1[dat[,i],],data1[-dat[,i],"y"],k=7)
#knn(训练集，测试集，训练集类标号，近邻数)
      test<-data1[dat[,i],"y"] #实际值
      biaoge[[4]]<-table(yy,test) #建立预测值与实际值的列联表
      typei[[4]]<-c(typei[[4]],biaoge[[4]][2,1]/sum(biaoge[[4]][,1]))

typeii[[4]]<-c(typeii[[4]],biaoge[[4]][1,2]/sum(biaoge[[4]][,2]))
      corrate4<-sum(diag(biaoge[[4]]))/sum(biaoge[[4]]) #对脚线元素之
和为正确分类的个数

result[[4]]<-c(result[[4]],corrate[[4]]<-sum(diag(biaoge[[4]]))/sum(b
iaoge[[4]])) #在交叉验证下求出各折的正确率

yy2<-knn(data1[-dat[,i],],data1[-dat[,i],],data1[-dat[,i],"y"],k=3)
#knn(训练集，测试集，训练集类标号，近邻数)
      test2<-data1[-dat[,i],"y"] #实际值
      biaoge2[[4]]<-table(yy2,test2) #建立预测值与实际值的列联表

typei2[[4]]<-c(typei2[[4]],biaoge2[[4]][2,1]/sum(biaoge2[[4]][,1]))

typeii2[[4]]<-c(typeii2[[4]],biaoge2[[4]][1,2]/sum(biaoge2[[4]][,2]))
      corrate2[[4]]<-sum(diag(biaoge2[[4]]))/sum(biaoge2[[4]]) #对脚
线元素之和为正确分类的个数

result2[[4]]<-c(result2[[4]],corrate2[[4]]<-sum(diag(biaoge2[[4]]))/s
um(biaoge2[[4]])) #在交叉验证下求出各折的正确率
    }

    #SVM
    for(i in 1:k){

svmtrain<-tune(svm,y~.,data=data1[-dat[,i],],kernel="radial",ranges=l
ist(cost=c(0.1,1,10,100,1000),gamma=c(0.1,0.5,1,2,3,4,5,7,7,8,9,10)))
#通过交叉验证选取合适的 cost 和 gamma 值
      yy<-predict(svmtrain$best.model,newdata=data1[dat[,i],]) #计算
预测值
      test<-data1[dat[,i],"y"] #实际值
      biaoge[[5]]<-table(yy,test) #建立预测值与实际值的列联表
      typei[[5]]<-c(typei[[5]],biaoge[[5]][2,1]/sum(biaoge[[5]][,1]))

typeii[[5]]<-c(typeii[[5]],biaoge[[5]][1,2]/sum(biaoge[[5]][,2]))
```

```
      corrate[[5]]<-sum(diag(biaoge[[5]]))/sum(biaoge[[5]]) #对脚线元
素之和为正确分类的个数

result[[5]]<-c(result[[5]],corrate[[5]]<-sum(diag(biaoge[[5]]))/sum(b
iaoge[[5]])) #在交叉验证下求出各折的正确率
      yy2<-predict(svmtrain$best.model,newdata=data1[-dat[,i],]) #计
算预测值
      test2<-data1[-dat[,i],"y"] #实际值
      biaoge2[[5]]<-table(yy2,test2) #建立预测值与实际值的列联表

typei2[[5]]<-c(typei2[[5]],biaoge2[[5]][2,1]/sum(biaoge2[[5]][,1]))

typeii2[[5]]<-c(typeii2[[5]],biaoge2[[5]][1,2]/sum(biaoge2[[5]][,2]))
      corrate2[[5]]<-sum(diag(biaoge2[[5]]))/sum(biaoge2[[5]]) #对脚
线元素之和为正确分类的个数

result2[[5]]<-c(result2[[5]],corrate2[[5]]<-sum(diag(biaoge2[[5]]))/s
um(biaoge2[[5]])) #在交叉验证下求出各折的正确率
    }

    #Logistic 回归
    for(i in 1:k){
      glmtrain<-glm(y~.,data1[-dat[,i],],family=binomial) #要设定
familiy 的值

glmpred<-predict(glmtrain,newdata=data1[dat[,i],],type="response") #
求出 y=1 的概率
      yy<-ifelse(glmpred>.5,1,0) #概率值大于 0.5 时认为属于第 1 类
      test<-data1[dat[,i],"y"]
      biaoge[[6]]<-table(yy,test)
      typei[[6]]<-c(typei[[6]],biaoge[[6]][2,1]/sum(biaoge[[6]][,1]))

typeii[[6]]<-c(typeii[[6]],biaoge[[6]][1,2]/sum(biaoge[[6]][,2]))
      corrate[[6]]<-sum(diag(biaoge[[6]]))/sum(biaoge[[6]])

result[[6]]<-c(result[[6]],corrate[[6]]<-sum(diag(biaoge[[6]]))/sum(b
iaoge[[6]]))

glmpred2<-predict(glmtrain,newdata=data1[-dat[,i],],type="response")
#求出 y=1 的概率
      yy2<-ifelse(glmpred2>.5,1,0) #概率值大于 0.5 时认为属于第 1 类
      test2<-data1[-dat[,i],"y"]
      biaoge2[[6]]<-table(yy2,test2)
```

```
typei2[[6]]<-c(typei2[[6]],biaoge2[[6]][2,1]/sum(biaoge2[[6]][,1]))

typeii2[[6]]<-c(typeii2[[6]],biaoge2[[6]][1,2]/sum(biaoge2[[6]][,2]))
      corrate2[[6]]<-sum(diag(biaoge2[[6]]))/sum(biaoge2[[6]])

result2[[6]]<-c(result2[[6]],corrate2[[6]]<-sum(diag(biaoge2[[6]]))/s
um(biaoge2[[6]]))
    }

    #线性判别分析 lda
    for(i in 1:k){
      ldatrain<-lda(y~.,data1[-dat[,i],]) #训练样本
      ldapred<-predict(ldatrain,newdata=data1[dat[,i],]) #计算预测值
      yy<-ldapred$class
      test<-data1[dat[,i],"y"] #实际值
      biaoge[[7]]<-table(yy,test) #建立预测值与实际值的列联表
      typei[[7]]<-c(typei[[7]],biaoge[[7]][2,1]/sum(biaoge[[7]][,1]))

typeii[[7]]<-c(typeii[[7]],biaoge[[7]][1,2]/sum(biaoge[[7]][,2]))
      corrate[[7]]<-sum(diag(biaoge[[7]]))/sum(biaoge[[7]]) #对脚线元
素之和为正确分类的个数

result[[7]]<-c(result[[7]],corrate[[7]]<-sum(diag(biaoge[[7]]))/sum(b
iaoge[[7]])) #在交叉验证下求出各折的正确率
      ldapred2<-predict(ldatrain,newdata=data1[-dat[,i],]) #计算预测
值
      yy2<-ldapred2$class
      test2<-data1[-dat[,i],"y"] #实际值
      biaoge2[[7]]<-table(yy2,test2) #建立预测值与实际值的列联表

typei2[[7]]<-c(typei2[[7]],biaoge2[[7]][2,1]/sum(biaoge2[[7]][,1]))

typeii2[[7]]<-c(typeii2[[7]],biaoge2[[7]][1,2]/sum(biaoge2[[7]][,2]))
      corrate2[[7]]<-sum(diag(biaoge2[[7]]))/sum(biaoge2[[7]]) #对脚
线元素之和为正确分类的个数

result2[[7]]<-c(result2[[7]],corrate2[[7]]<-sum(diag(biaoge2[[7]]))/s
um(biaoge2[[7]])) #在交叉验证下求出各折的正确率
    }

    #线性判别分析 qda
    for(i in 1:k){
      qdatrain<-qda(y~.,data1[-dat[,i],]) #训练样本
```

```
      qdapred<-predict(qdatrain,newdata=data1[dat[,i],]) #计算预测值
      yy<-qdapred$class
      test<-data1[dat[,i],"y"] #实际值
      biaoge[[8]]<-table(yy,test) #建立预测值与实际值的列联表
      typei[[8]]<-c(typei[[8]],biaoge[[8]][2,1]/sum(biaoge[[8]][,1]))

typeii[[8]]<-c(typeii[[8]],biaoge[[8]][1,2]/sum(biaoge[[8]][,2]))
      corrate[[8]]<-sum(diag(biaoge[[8]]))/sum(biaoge[[8]]) #对脚线元
素之和为正确分类的个数

result[[8]]<-c(result[[8]],corrate[[8]]<-sum(diag(biaoge[[8]]))/sum(b
iaoge[[8]])) #在交叉验证下求出各折的正确率
      qdapred2<-predict(qdatrain,newdata=data1[-dat[,i],]) #计算预测
值
      yy2<-qdapred2$class
      test2<-data1[-dat[,i],"y"] #实际值
      biaoge2[[8]]<-table(yy2,test2) #建立预测值与实际值的列联表

typei2[[8]]<-c(typei2[[8]],biaoge2[[8]][2,1]/sum(biaoge2[[8]][,1]))

typeii2[[8]]<-c(typeii2[[8]],biaoge2[[8]][1,2]/sum(biaoge2[[8]][,2]))
      corrate2[[8]]<-sum(diag(biaoge2[[8]]))/sum(biaoge2[[8]]) #对脚
线元素之和为正确分类的个数

result2[[8]]<-c(result2[[8]],corrate2[[8]]<-sum(diag(biaoge2[[8]]))/s
um(biaoge2[[8]])) #在交叉验证下求出各折的正确率
    }

    #Probit 回归
    for(i in 1:k){

glmtrain<-glm(y~.,data1[-dat[,i],],family=binomial(link="probit")) #
要设定 familiy 的值

glmpred<-predict(glmtrain,newdata=data1[dat[,i],],type="response") #
求出 y=1 的概率
      yy<-ifelse(glmpred>.5,1,0) #概率值大于 0.5 时认为属于第 1 类
      test<-data1[dat[,i],"y"]
      biaoge[[9]]<-table(yy,test) #建立预测值与实际值的列联表
      typei[[9]]<-c(typei[[9]],biaoge[[9]][2,1]/sum(biaoge[[9]][,1]))

typeii[[9]]<-c(typeii[[9]],biaoge[[9]][1,2]/sum(biaoge[[9]][,2]))
      corrate[[9]]<-sum(diag(biaoge[[9]]))/sum(biaoge[[9]]) #对脚线元
素之和为正确分类的个数
```

```
result[[9]]<-c(result[[9]],corrate[[9]]<-sum(diag(biaoge[[9]]))/sum(biaoge[[9]])) #在交叉验证下求出各折的正确率

glmpred2<-predict(glmtrain,newdata=data1[-dat[,i],],type="response") #求出 y=1 的概率
      yy2<-ifelse(glmpred2>.5,1,0) #概率值大于 0.5 时认为属于第 1 类
      test2<-data1[-dat[,i],"y"]
      biaoge2[[9]]<-table(yy2,test2) #建立预测值与实际值的列联表

typei2[[9]]<-c(typei2[[9]],biaoge2[[9]][2,1]/sum(biaoge2[[9]][,1]))

typeii2[[9]]<-c(typeii2[[9]],biaoge2[[9]][1,2]/sum(biaoge2[[9]][,2]))
      corrate2[[9]]<-sum(diag(biaoge2[[9]]))/sum(biaoge2[[9]]) #对脚线元素之和为正确分类的个数

result2[[9]]<-c(result2[[9]],corrate2[[9]]<-sum(diag(biaoge2[[9]]))/sum(biaoge2[[9]])) #在交叉验证下求出各折的正确率
    }

    #Boosting
    for(i in 1:k){
      boostingtrain<-boosting(y~.,data1[-dat[,i],]) #训练树
      boostingpred<-predict(boostingtrain,newdata=data1[dat[,i],]) #计算预测值
      yy<-boostingpred$class
      test<-data1[dat[,i],"y"] #实际值
      biaoge[[10]]<-table(yy,test) #建立预测值与实际值的列联表

typei[[10]]<-c(typei[[10]],biaoge[[10]][2,1]/sum(biaoge[[10]][,1]))

typeii[[10]]<-c(typeii[[10]],biaoge[[10]][1,2]/sum(biaoge[[10]][,2]))
      corrate[[10]]<-sum(diag(biaoge[[10]]))/sum(biaoge[[10]]) #对脚线元素之和为正确分类的个数

result[[10]]<-c(result[[10]],corrate[[10]]<-sum(diag(biaoge[[10]]))/sum(biaoge[[10]])) #在交叉验证下求出各折的正确率
      boostingpred2<-predict(boostingtrain,newdata=data1[-dat[,i],]) #计算预测值
      yy2<-boostingpred2$class
      test2<-data1[-dat[,i],"y"] #实际值
      biaoge2[[10]]<-table(yy2,test2) #建立预测值与实际值的列联表
```

```
typei2[[10]]<-c(typei2[[10]],biaoge2[[10]][2,1]/sum(biaoge2[[10]][,1]
))

typeii2[[10]]<-c(typeii2[[10]],biaoge2[[10]][1,2]/sum(biaoge2[[10]][,
2]))
      corrate2[[10]]<-sum(diag(biaoge2[[10]]))/sum(biaoge2[[10]]) #对
脚线元素之和为正确分类的个数

result2[[10]]<-c(result2[[10]],corrate2[[10]]<-sum(diag(biaoge2[[10]]
))/sum(biaoge2[[10]])) #在交叉验证下求出各折的正确率
    }

    #BP 人工神经网络
    for(i in 1:k){
      bb<-class.ind(data1$y)#生成输出值标号矩阵

nnettrain<-nnet(data1[-dat[,i],-1],bb[-dat[,i],],size=3,rang=0.1,deca
y=5e-4,maxit=200) #训练网络 nnet(X,Y,...)
      nnetpred<-predict(nnettrain,data1[dat[,i],-1]) #计算预测值
      yy<-max.col(nnetpred)
      test<-max.col(bb[dat[,i],])
      biaoge[[11]]<- table(yy,test) #建立预测值与实际值的列联表

typei[[11]]<-if(nrow(biaoge[[11]])==1){c(typei[[11]],0)}else{c(typei[
[11]],biaoge[[11]][2,1]/sum(biaoge[[11]][,1]))}

typeii[[11]]<-if(ncol(biaoge[[11]])==1){c(typeii[[11]],0)}else{c(type
ii[[11]],biaoge[[11]][1,2]/sum(biaoge[[11]][,2]))}
      corrate[[11]]<-sum(diag(biaoge[[11]]))/sum(biaoge[[11]]) #对脚
线元素之和为正确分类的个数

result[[11]]<-c(result[[11]],corrate[[11]]<-sum(diag(biaoge[[11]]))/s
um(biaoge[[11]])) #在交叉验证下求出各折的正确率
      nnetpred2<-predict(nnettrain,data1[-dat[,i],-1]) #计算预测值
      yy2<-max.col(nnetpred2)
      test2<-max.col(bb[-dat[,i],]) #实际值
      biaoge2[[11]]<-table(yy2,test2) #建立预测值与实际值的列联表

typei2[[11]]<-if(nrow(biaoge2[[11]])==1){c(typei2[[11]],0)}else{c(typ
ei2[[11]],biaoge2[[11]][2,1]/sum(biaoge2[[11]][,1]))}

typeii2[[11]]<-if(ncol(biaoge2[[11]])==1){c(typeii2[[11]],0)}else{c(t
ypeii2[[11]],biaoge2[[11]][1,2]/sum(biaoge2[[11]][,2]))}
```

```
      corrate2[[11]]<-sum(diag(biaoge2[[11]]))/sum(biaoge2[[11]]) #对
脚线元素之和为正确分类的个数

result2[[11]]<-c(result2[[11]],corrate2[[11]]<-sum(diag(biaoge2[[11]]
))/sum(biaoge2[[11]])) #在交叉验证下求出各折的正确率
    }

    #生成预测值结果表格
    result<-as.data.frame(result) #把 list 形式转换为数据框
    colnames(result)<-c("决策树","Bagging","随机森林","K 近邻
","SVM","Logistic 回归","线性判别分析 lda","线性判别分析 qda","Probit
回归","Boosting","BP 神经网络") #重新命名列标题
    rownames(result)<-1:k #重新命名列标题


    #生成拟合值结果表格
    result2<-as.data.frame(result2) #把 list 形式转换为数据框
    colnames(result2)<-c("决策树","Bagging","随机森林","K 近邻
","SVM","Logistic 回归","线性判别分析 lda","线性判别分析 qda","Probit
回归","Boosting","BP 神经网络") #重新命名列标题
    rownames(result2)<-1:k #重新命名列标题

    #生成预测值第一类错误率结果表格
    typei<-as.data.frame(typei) #把 list 形式转换为数据框
    colnames(typei)<-c("决策树","Bagging","随机森林","K 近邻
","SVM","Logistic 回归","线性判别分析 lda","线性判别分析 qda","Probit
回归","Boosting","BP 神经网络") #重新命名列标题
    rownames(typei)<-1:k #重新命名列标题

    #生成预测值第二类错误率结果表格
    typeii<-as.data.frame(typeii) #把 list 形式转换为数据框
    colnames(typeii)<-c("决策树","Bagging","随机森林","K 近邻
","SVM","Logistic 回归","线性判别分析 lda","线性判别分析 qda","Probit
回归","Boosting","BP 神经网络") #重新命名列标题
    rownames(typeii)<-1:k #重新命名列标题

    #生成训练值第一类错误率结果表格
    typei2<-as.data.frame(typei2) #把 list 形式转换为数据框
    colnames(typei2)<-c("决策树","Bagging","随机森林","K 近邻
","SVM","Logistic 回归","线性判别分析 lda","线性判别分析 qda","Probit
回归","Boosting","BP 神经网络") #重新命名列标题
    rownames(typei2)<-1:k #重新命名列标题

    #生成训练值第二类错误率结果表格
```

```
    typeii2<-as.data.frame(typeii2) #把 list 形式转换为数据框
    colnames(typeii2)<-c("决策树","Bagging","随机森林","K 近邻
","SVM","Logistic 回归","线性判别分析 lda","线性判别分析 qda","Probit
回归","Boosting","BP 神经网络") #重新命名列标题
    rownames(typeii2)<-1:k #重新命名列标题


    #生成结果的统计描述
    stats<-function(dafa){
      c(平均值=mean(dafa),标准差=sd(dafa),最小值=min(dafa),下四分位数
=quantile(dafa,0.25),中位数=median(dafa),上四分位数
=quantile(dafa,0.75),最大值=max(dafa))
    }
    resultstats<-apply(result,2,stats)
    resultstats2<-apply(result2,2,stats)
    typeistats<-apply(typei,2,stats)
    typeiistats<-apply(typeii,2,stats)
    typeistats2<-apply(typei2,2,stats)
    typeiistats2<-apply(typeii2,2,stats)
    colnames(resultstats)<-colnames(result) #重新命名行标题
    colnames(resultstats2)<-colnames(result2)
    colnames(typeistats)<-colnames(typei)
    colnames(typeiistats)<-colnames(typeii)
    colnames(typeistats2)<-colnames(typei2)
    colnames(typeiistats2)<-colnames(typeii2)
```