中国研究生创新实践系列大赛

“华为杯”第十八届中国研究生

数学建模竞赛

学　　校	上海师范大学
参赛队号	**21102700119**
队员姓名	**1. 陆悦**
	2. 熊斯洁
	3. 施晨扬

中国研究生创新实践系列大赛

“华为杯”第十八届中国研究生

数学建模竞赛

题 目 基于数据挖掘的抗乳腺癌候选药物的优化模型

摘 要：

乳腺癌是目前世界上常见的高致死率癌症之一，严重威胁了人类的生命健康。抗癌药物的筛选具有重要研究意义和价值。本文基于数据挖掘和机器学习技术，研究抗癌药物筛的优化建模问题，具有一定的现实意义。

针对问题一，首先对数据进行了预处理，剔除数据中存在的缺失值、异常值、异常变量和异常样本。在变量筛选过程中，考虑了变量之间的线性和非线性关系。首先，采用 LASSO 回归、person 相关系数、随机森林和互信息 4 种方法分别得到排序前 40 的变量。然后，采用投票加权的方式，找到综合排序前 40 的变量子集。最后，通过高相关滤波方法向后迭代去除相关性较高的变量，得到最终的 20 个最优变量，保证提取的变量具有代表性和独立性。

针对问题二，首先基于筛选的 20 个变量建立了基于直方图的梯度增强回归树的回归预测模型 HGBRT。在训练过程中将 1973 个样本按照 8:2 分为训练集和测试集，采用 K-折交叉验证方法对模型在训练集进行预训练，在测试集上完成测试与验证。然后，将该模型与多种经典的回归模型如：支持向量回归 SVR、神经网络 MLP 和随机森林 RF 等方法进行对比验证，依据算法在测试集上的预测误差表明采用的 HGBRT 具有最小的误差和更好回归预测效果。最后，通过网格搜所对模型的超参数寻优，确定了模型最优超参数。

针对问题三，首先，采用问题一的方法分别筛选针对化合物 Caco-2、CYP3A4、hERG、HOB 和 MN5 个因变量相关的 20 个最优变量。然后，基于筛选的变量建立了基于直方图的梯度增强分类树的分类预测模型 HGBCT。在训练过程中，考虑到 5 个化合物分类中 0-1 样本分布不均衡可能对模型分类精度影响，通过增加对小样本分类错误的惩罚因子 C 来解决该问题。通过在测试集上与多种分类模型对比，发现 HGBCT 具有较高的分类精度，同时随机森林方法也具有近似的分类精度。最后，通过调节惩罚因子 C 可以得到当惩罚因此 C 取值为 3~5 之间时，HGBCT 分类准确率最高。

针对问题四，首先基于问题二和问题三提取的相关变量进行投票加权选择评分较高的变量作为优化变量集。变量集中的变量对 ERα 生物活性和 ADMET 药性具有联合的影响。然后，基于 HGBRT 值和 HGBCT 构建双目标优化模型，以函数取值最大为目标函数，变量的给定范围为约束，求解使得目标函数最大值时变量的取值；考虑到活性因素和药性因素占比，增加权重参数 λ。模型求解采用第三代非支配排序遗传算法(NSGA-III)进行启发式寻优。最后，通过对算法的初始化参数进行调试，初步确定了 NSGA-III 比较优化的初始化参数，并给出权重参数 λ=0.5 时的一组最优解。经过验证，模型求解结果符合要求。同时，对权重 λ 的占比进行求解发现 λ 取值在 0.5~0.6 之间，模型可以取得最大值，该区间可以作为参数优化的参考区间。

关键词：相关性分析，HGBRT，HGBCT，NSGA-III，机器学习，数据挖掘

目录

1. 问题重述

1.1 问题背景

乳腺癌是世界上目前常见的高致死率癌症之一，该病患者中女性占 99%，全世界每年约有 120 万妇女罹患乳腺癌,有 50 万妇女死于此病。目前，乳腺癌的发病率呈不断升高的趋势，已成为全球医学界所瞩目的问题[1]。乳腺癌的发展与雌激素受体密切相关，雌激素受体 α 亚型在不超过 10%的正常乳腺上皮细胞中表达，但大约在 50%-80%的乳腺肿瘤细胞中表达；在对 ERα 基因缺失小鼠的实验结果表明，ERα 确实在乳腺发育过程中扮演了十分重要的角色。目前，抗激素治疗常用于 ERα 表达的乳腺癌患者，其通过调节雌激素受体活性来控制体内雌激素水平。因此，ERα 被认为是治疗乳腺癌的重要靶标，能够拮抗 ERα 活性的化合物可能是治疗乳腺癌的候选药物。

在药物研发中，为了节约时间和成本，通常采用建立化合物活性预测模型的方法来筛选潜在活性化合物。针对与疾病相关的某个靶标，收集一系列作用于该靶标的化合物及其生物活性数据，然后以一系列分子结构描述符作为自变量，化合物的生物活性值作为因变量，构建化合物的定量结构-活性关系模型，然后使用该模型预测具有更好生物活性的新化合物分子，或者指导已有活性化合物的结构优化。

一个化合物想要成为候选药物，除了需要具备良好的生物活性（此处指抗乳腺癌活性）外，还需要在人体内具备良好的药代动力学性质和安全性，合称为 ADMET（Absorption 吸收、Distribution 分布、Metabolism 代谢、Excretion 排泄、Toxicity 毒性）性质。其中，ADME 主要指化合物的药代动力学性质，描述了化合物在生物体内的浓度随时间变化的规律，T 主要指化合物可能在人体内产生的毒副作用。一个化合物的活性再好，如果其 ADMET 性质不佳，比如很难被人体吸收，或者体内代谢速度太快，或者具有某种毒性，那么其仍然难以成为药物，因而还需要进行 ADMET 性质优化。本题仅考虑化合物的 5 种 ADMET 性质：小肠上皮细胞渗透性（Caco-2）；细胞色素 P450 酶（Cytochrome P450, CYP）3A4 亚型（CYP3A4）；人体口服生物利用度（Human Oral Bioavailability, HOB）；微核试验（Micronucleus，MN）。

本试题针对乳腺癌治疗靶标 ERα 药物筛选问题进行建模研究，提供了 ERα 拮抗剂信息（1974 个化合物样本，每个样本都有 729 个分子描述符变量，1 个生物活性数据，5 个 ADMET 性质数据），构建化合物生物活性的定量预测模型和 ADMET 性质的分类预测模型，从而为同时优化 ERα 拮抗剂的生物活性和 ADMET 性质提供预测服务。本试题提供四个表格数据，分别为“ERα_activity.xlsx”，“Molecular_Descriptor.xlsx”，“ADMET.xlsx”，“分子描述符含义解释.xlsx”。

“ERα_activity.xlsx”包括训练集和测试集两个子表格。训练集提供了 1974 个化合物对 ERα 的生物活性数据，在本文件中第一列表示 1974 个化合物的结构式；第二列是化合物对 ERα 的生物活性值；第三列是将第二列 IC_{50} 值转化而得的 pIC_{50}。测试集给出了 50 个分子式，需要对其生物特性数据进行填充。

“Molecular_Descriptor.xlsx”给出了 1974 个化合物的 729 个分子描述符信息，每列代表化合物的一个自变量。化合物的分子描述符是一系列用于描述化合物的结构和性质特征的参数，每个变量的具体含义可以见“分子描述符含义解释.xlsx”的详细描述。“Molecular_Descriptor.xlsx”包含了训练集和测试集分别用于模型训练和解答。

“ADMET.xlsx”给出了包含上述 1974 个化合物的 5 种 ADMET 性质的数据。其中第

一列也是表示化合物结构的 SMILES 式，并采用二分类法提供相应的取值。同时该表格也包含训练集和测试集用于模型训练和求解。

1.2 问题解决

基于上述研究背景和题目所给数据表需要解决以下问题：

问题一：数据降维与特征选择

针对 1974 个化合物的 729 个分子描述符进行变量选择，根据变量对生物活性影响的重要性进行排序，并给出前 20 个对生物活性最具有显著影响的分子描述符（即变量），并请详细说明分子描述符筛选过程及其合理性。

问题二：回归预测建模

选择不超过 20 个分子描述符变量，构建化合物对 ERα 生物活性的定量预测模型，对文件“ERα_activity.xlsx”的 test 表中的 50 个化合物进行 IC_{50} 值和对应的 pIC_{50} 值预测。

问题三：分类预测模型

请利用文件“Molecular_Descriptor.xlsx”提供的 729 个分子描述符，针对文件“ADMET.xlsx”中提供的 1974 个化合物的 ADMET 数据，分别构建化合物的 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型，并简要叙述建模过程。使用所构建的 5 个分类预测模型，对文件“ADMET.xlsx”的 test 表中的 50 个化合物进行相应的预测，并将结果填入“ADMET.xlsx”的 test 表中对应的 Caco-2、CYP3A4、hERG、HOB、MN 列。

问题四：优化模型

并阐述化合物的哪些分子描述符，以及这些分子描述符在什么取值或者处于什么取值范围时，能够使化合物对抑制 ERα 具有更好的生物活性，同时具有更好的 ADMET 性质。（给定的五个 ADMET 性质中，至少三个性质较好）。

2 符号说明

3.1 符号说明

符号的相关含义解释在文中均有注明，以下展示部分符号变量

序号	符号	含义
1	h_i	第 i 个样本
2	H	样本集
3	a_j	第 j 个变量
4	A	变量集
5	MSE	均方误差
6	RMSE	均方根误差
7	R^2	R^2 可决系数
8	Acc	分类准确率

3 模型假设

1）假设实验数据来源真实可靠，数据噪声误差满足在合理范围；
2）假设化合物理化性质相对稳定，每个化合物理化性质相对独立互不影响；
3）假设每个化合物的分子变量数值都单独测量，实验数据满足测试规范。

4 问题一：变量选择

4.1 问题一分析

本题为数据分析建模问题，在进行特征选择和建模之前要进行数据的预处理。数据预处理主要针对“Molecular_Descriptor.xlsx”表格中 1974 个化合物的 729 个变量的取值进行数据清洗，主要查找是否有缺失值、异常值等。数据预处理可以简化后续变脸提取过程。

对于缺失值主要有删除和插补处理，由于本题数据为实验测量值，每个化合物都有单独的性质并且变量没有前后时序关系，并且经过查找，数据表中不存在缺失值。

异常值去除。通过对变量进行初步统计特征分析，其中很多变量存在大部分或者全部为 0 值，说明该份子描述符对分子活性没有影响可以直接删除。如果某个化合物变量的某个变量存在突变，并且该突变是否产生了明显的分子活性，则不应删除。

针对数据降维问题，需要对化合物的 729 个变量进行排序，给出前 20 个对生物活性具有显著影响的变量，并说明变量筛选过程与合理性。对于变量与生物活性值之间可能存在线性和非线性关系。常用的线性降维方法主要有 LASSO 和 Pearson，非线性降维主要基于机器学习的随机森林、XGBoost 等方法。但是每种方法必然得到变量排序结果必然是不同的。本问题的难点在于：（1）分子描述符变量数量较多，并且变量与生物活性之间的线性和非线性关系难以直接确定。（2）变量数量较多，变量之间可能存在高度耦合的关联关系，需要对选择出变量的独立性问题进行讨论

针对难点（1）的代表性变脸选择问题。由于特征变量选择是为了后续更好的建立模型并且预先无法知道变量与生物活性之间的相关关系，本文采用多种数据降维方法首先得到前 50 个变量排序变量子集，然后通过投票的方式，找到最优的 20 个变量。

针对难点（2）的高度耦合变量之间的独立性问题，采用高相关性滤波对变量的相关关系进行独立性检验。对于存在高度相关性的变量，可以舍去其中排序较后的变量，然后在子集中继续选择，重复以上过程，直到选择出 20 个具有独立性的变量。

原始数据为 1974×729 的数值矩阵，其中每个化合物样本记为 $h_i \in H,(i=1,2,\cdots,1974)$，组成样本集。每个分子描述符为变量记为 $a_j \in A,(j=1,2,\cdots,729)$，组成变量集。变量值记为 x_{ij}，表示第 i 个样本的第 j 个变量的数值。因变量生物活性记为 $y_i \in Y,(i=1,2,\cdots,1974)$ 表示第 i 个化合物的生物活性值。本问题的目标是选择前 20 个变量组成新的变量集。

注：变量的序号与“Molecular_Descriptor.xlsx”表格中对应的顺序相差为 1，数据编号从第一个样本和变量值算起与行列号差值为 1.

题目一的整个思路流程如下图所示：

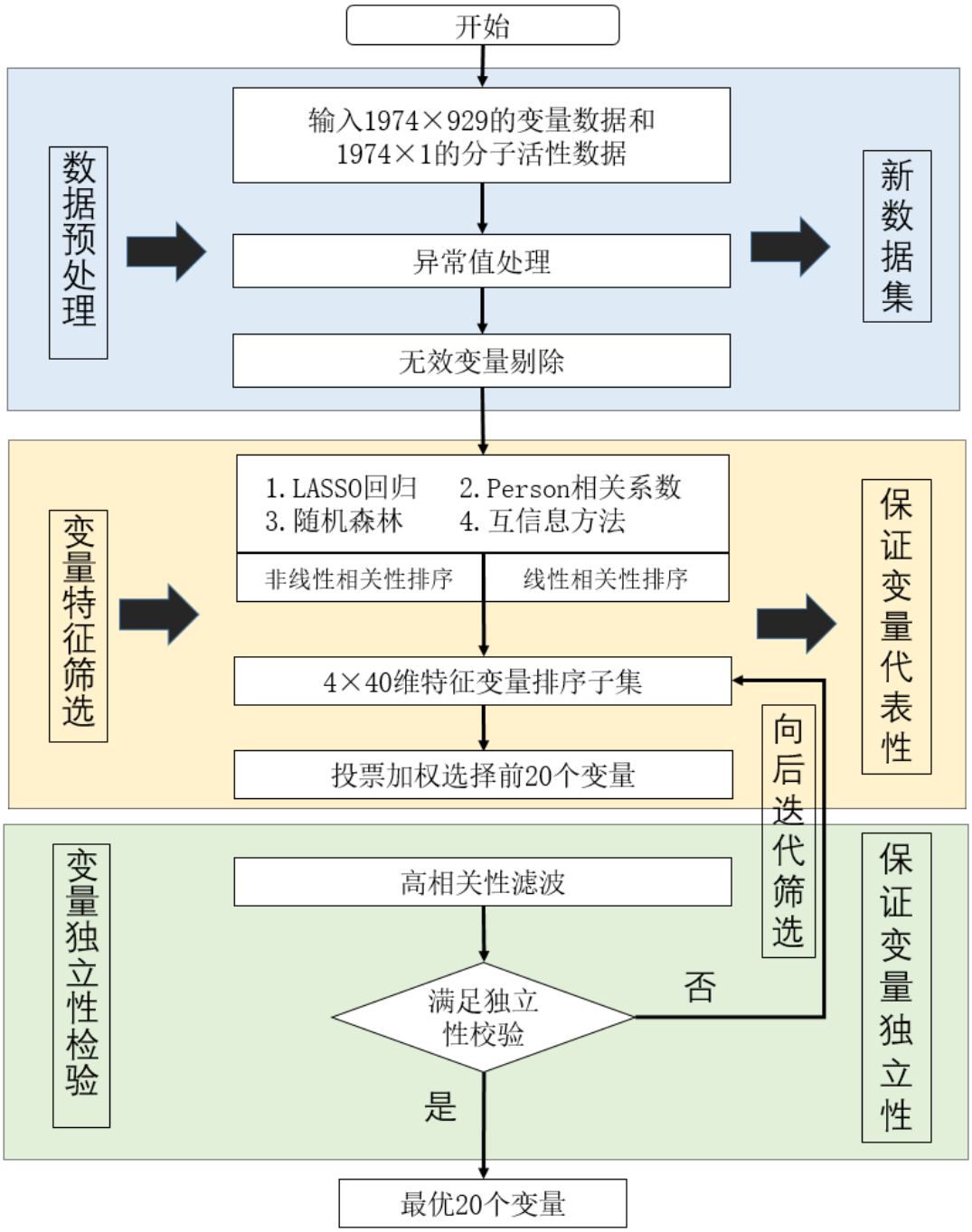


图 4.1 问题一建模思路分析流程图

4.2 数据预处理

1）缺失值处理

首先，利用 python 的 pandas 对“Molecular_Descriptor.xlsx”进行数据分析，发现没有缺失值，因此不用数据进行缺失值处理。

2）异常值处理

通常情况下进行数据分析的数据样本应尽量满足一定的分布，如常见的高斯分布等。但是由于是实验误差或者版本不合理可能导致出现与样本数据分布差别较大的值，这些值会造成数据回归或者拟合时出现较大的误差，为了避免距离数据分布中心较远值对模型的

影响，需要对异常值进行剔除。

经过数据分析和处理发现，对于第 1563 个化合物中大部分变量值存在较大的突变。因此，经核查第 1563 个化合物分子式如下：

CC[C@H](C)[C@@H]1NC(=O)[C@@H](CSSC[C@H](NC(=O)[C@H](CC(C)C)NC1=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)N)C(=O)NCC(=O)NCC(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N)NC(=O)[C@@H](N)CCCCN。

该分子式要比其他绝大部分所有化合物分子式都大，因此部分理化性质存在较大值为合理值。然而该样本的生物活性值 pIC50=6.125 并不突出，同时该样本的 ADMET 存在心脏毒性和遗传毒性等。无论从数据角度还是理化性质方面该样本都不是好的样本，因此将其进行删除。

3）多 0 值剔除

由于一些变量的所有值或者大部分值为 0，也说明该变量对生物活性没有或者基本没有影响，可以直接删除。本文选择删除样本变量中 0 值数量占比大于 95%的变量。经过编程统计共筛选出 225 个变量 0 值数量占比大于 95%的变量。

至此，得到的处理后的数据集样本满足后续特征筛选和建模的需求。共计剔除一个样本(1563)和 225 个变量。得到新的数据集（1973×504）进行后续特征筛选。

4.3 变量筛选——投票加权法

变量的筛选过程就是对变量进行数据降维，找到对生物活性最具有显著影响的变量。用 20 个变量可以近似可以表示出全部变量的生物活性指标。变量之间的影响可以用相关性表示，相关性越强则影响越显著。相关性有线性相关性和非线性相关性，而非线性关系要比线性关系复杂的多且更加难以描述。目前，单一特征筛选方法难以准确描述出变量之间的所有相关关系，并且每一种方法衡量相关性的指标各不相同，因此我们考虑利用四种包含线性和非线性特征筛选方法的进行综合选择，综合考虑四种分析方法下变量相关性排序选出最优的特征变量，建立以下特征筛选模型：

假设方法 1~4 选择出的变量子集分别为 A1，A2，A3 和 A4，每个子集变量的数量为 40，并且已经按照相关性重要程度完成了排序。最优变量选择的指标是：（1）变量出现的次数（出现次数越多越好）（2）变量的排序（变量排序越前越好）的综合因素。由于每种方法评价的量纲不同，本文采用人工赋值投票的模型进行选择。

对于变量子集而言，每个变量按照出现的先后次序赋值为 50~11，每个变量值依次递减 1，即排序第一的变量赋值为 50，排序最后的变量赋值为 11。然后将所有出现的变量赋值求和得到所有变量子集的集合 A5，对 A5 中变量进行排序，最终前 20 个变量即为最终的选择结果。整个模型可以用流程图表示：

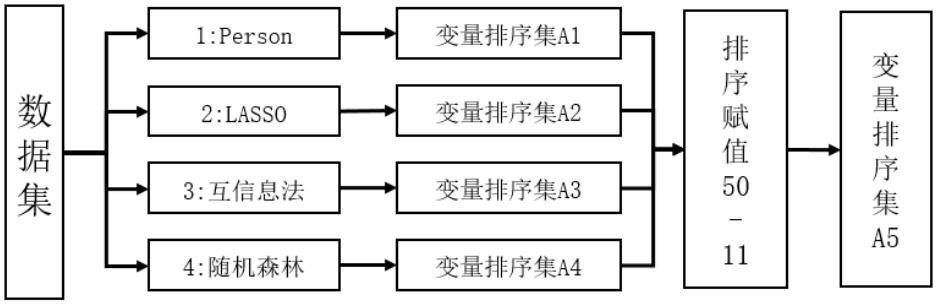


图 4.2 变量选择综合排序模型流程图

首先，可以对于采用相关系数分析，考察两个变量之间的相关程度。对于两个随机变量：X 和 Y，最终计算出的相关系数的含义可以如下解释：

（1）当相关系数为 0 时，X 和 Y 两个变量无关。

（2）当 X 的值增大（减小），Y 也增大（减小）时，两个变量正相关，相关系数在 0~1 之间。

（3）当 X 的值增加（减小）,Y 也减小（增大）时，两个变量负相关，相关系数在-1~0 之间。

相关系数的绝对值越大，相关性越强；相关系数越近于 0，相关系数越弱。通常情况下可以通过下表取值范围判断变量的相关程度：

表 4-1 系数相关性判别

相关系数	相关强度
0.8-1.0	极强相关
0.6-0.8	强相关
0.4-0.6	中等程度相关
0.2-0.4	弱相关
0.0-0.2	极弱相关或无相关

在相关系数方法中有皮尔森（pearson）相关系数和斯皮尔曼（spearman）相关系数和肯德尔（kendall）相关系数.皮尔森相关系数是衡量线性关联性的程度，描述变量与因变量之间的线性相关关系，本文选择 Person 相关系数进行相关性分析。

1. Person 相关系数分析

Person 相关系数输出范围为-1 到+1, 0 代表无相关性，负值为负相关，正值为正相关

$$\rho(X,Y)=\frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}=\frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{\sum_{i=1}^{n}(X_i-\mu_X)^2}\sqrt{\sum_{i=1}^{n}(Y_i-\mu_Y)^2}}$$

Person 相关系数得到的结果排序子集 A1 如图所示:

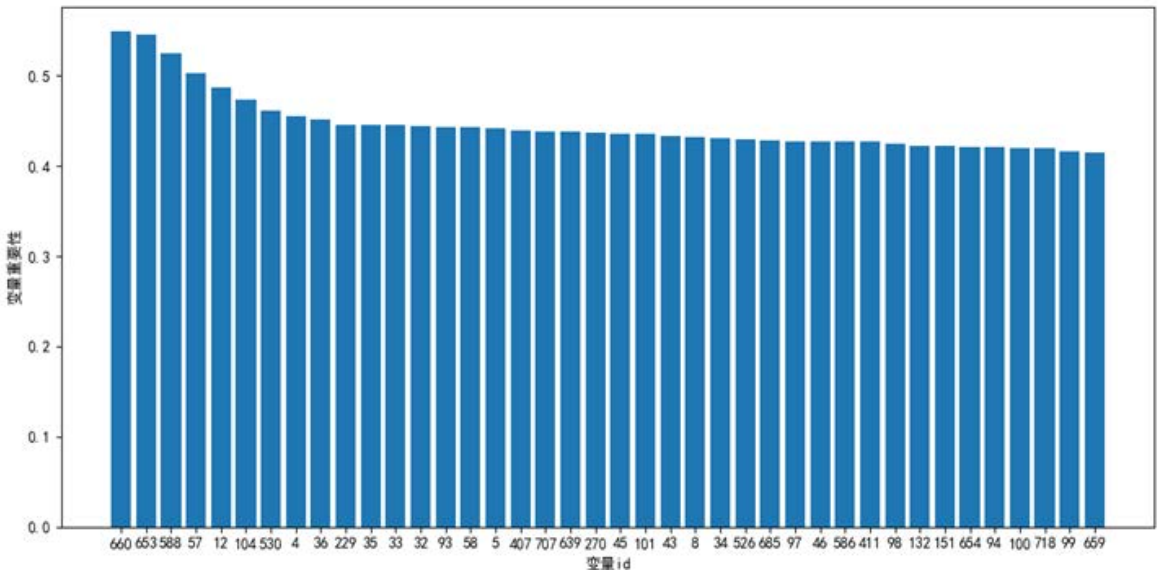


图 4.3 Person 相关分析得到的变量子集 A1 排序

2. LASSO

LASSO 是一种采用 L1 正则化的线性回归方法，其特点是在拟合广义线性模型的同时进行变量的筛序和复杂度调整。LASSO 变量筛选的优势在于不把所有的变量都放入模型中进行拟合，而是有选择的把变量放入模型从而得到更好的性能参数。LASSO 回归复杂度调整的程度由参数 λ 来控制，λ 越大对变量较多的线性模型的惩罚力度就越大，从而最终获得一个变量较少的模型，从而避免过度拟合。LASSO 回归方法得到变量子集排序为

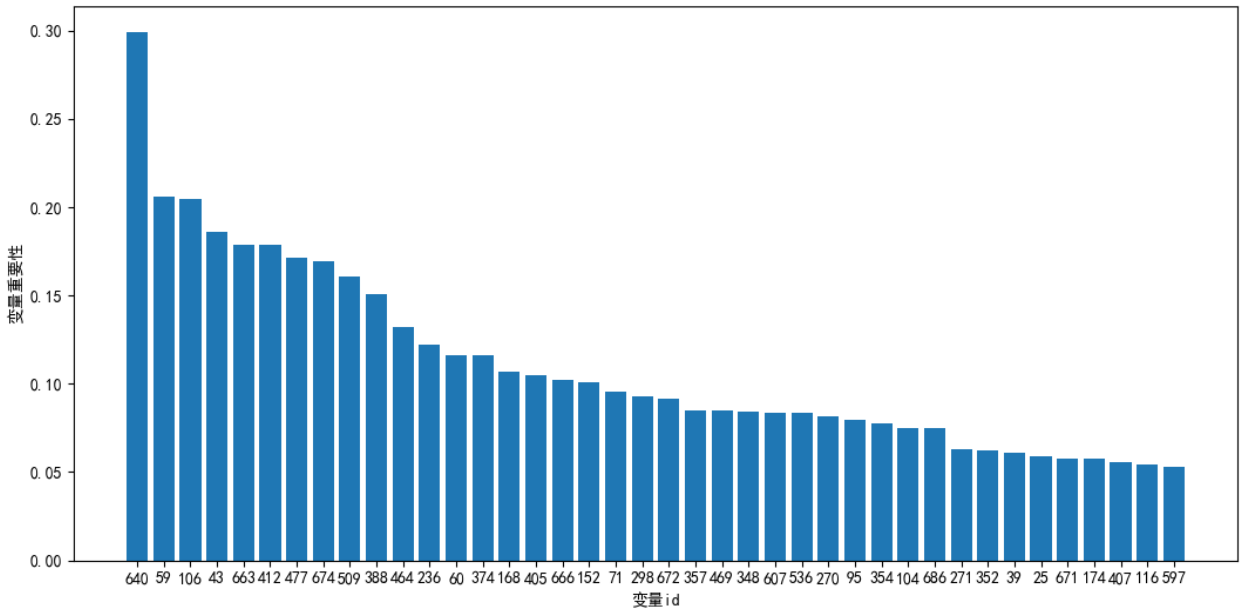


图 4.4 LASSO 回归分析得到的变量子集 A2 排序

3. 互信息法

在概率论和信息论中，两个随机变量的互信息或转移信息是是变量间相互依赖性的量度。不同于相关系数，互信息并不局限于实值随机变量，它更加一般且决定着联合分布 p(X,Y) 和分解的边缘分布的乘积 p(X)p(Y) 的相似程度。互信息是度量两个事件的集合之间的相关性。两个离散随机变量 X 和 Y 的互信息可以定义为：

$$I(X;Y)=\sum_{y\in Y}\sum_{x\in X}p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

互信息分析得到的前 40 个变量子集排序为

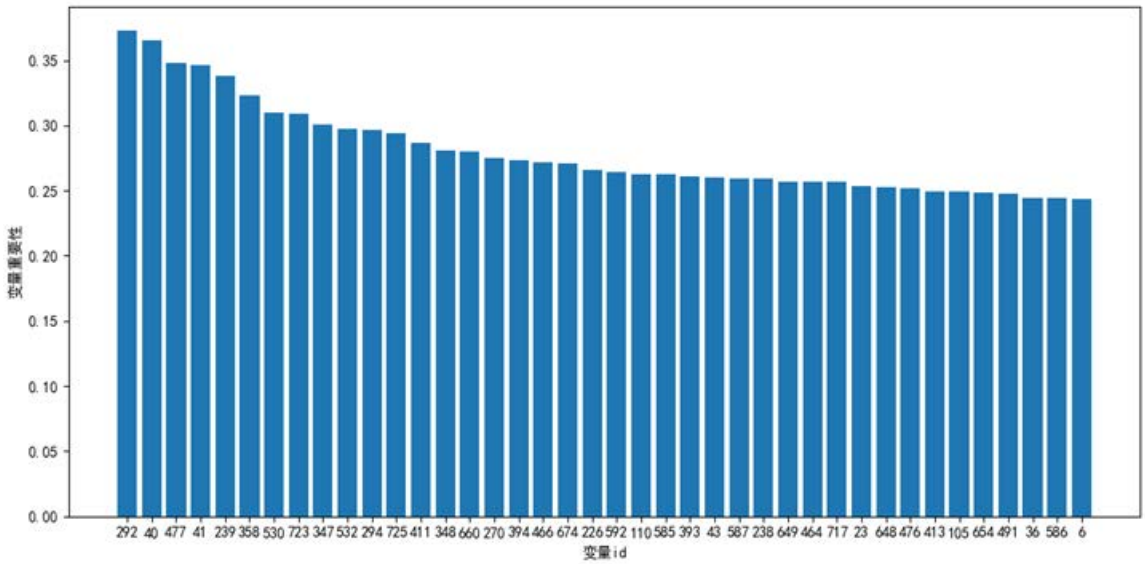


图 4.5 互信息分析得到的变量子集 A3 排序

4.随机森林

随机森林是基于 Bagging 思想对 N 个决策树为基分类器进行集成学习后得到的组合机器学习分类器。随机森林可以实现回归和分类分析，取决于随机森林是分类树还是回归树。如果采用的基分类器为回归树，采用最小均方差原则。对于任意划分特征 A，对应的任意划分点 s 两边划分成的数据集 D1 和 D2，求出使 D1 和 D2 各自集合的均方差最小，同时 D1 和 D2 的均方差之和最小所对应的特征和特征值划分点。表达式为：

$$\underbrace{min}_{A,s}\left[\underbrace{min}_{c_1}\sum_{x_i\in D_1(A,s)}(y_i-c_1)^2+\underbrace{min}_{c_2}\sum_{x_i\in D_2(A,s)}(y_i-c_2)^2\right]$$

其中，c1 为 D1 数据集的样本输出均值，c2 为 D2 数据集的样本输出均值。cart 树的预测是根据叶子结点的均值，因此随机森林的预测是所有树的预测值的平均值。随机森林可以实现复杂的非线性回归，同时具有很强的通用性和强大的拟合能力。随机森林还可以在拟合是获得每个变量的重要性排序，如下图所示：

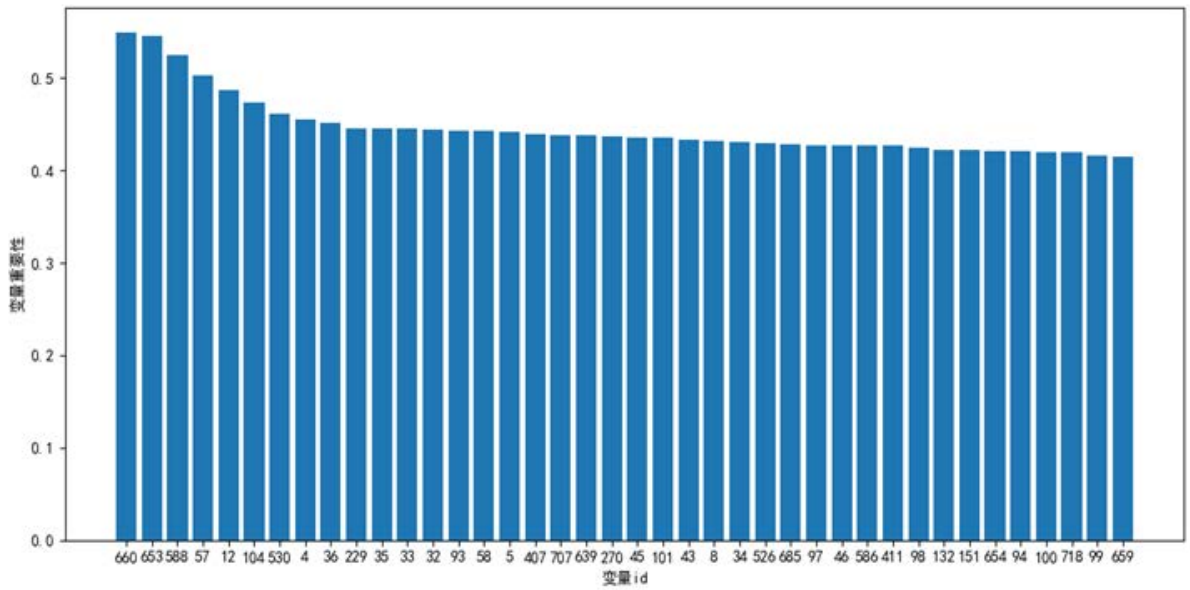


图 4.6 随机森林得到的变量子集 A4 排序

5. 投票加权排序

获得了 4 种方法的变量子集后，我们发现每种方法得到的特征子集各不相同，但是其中不乏存在一些变量出现了多次，如变量 a_{477} 和变量 a_{660}。但是需要用定量方法对变量完成排序。每个变量子集依次按照先后顺序对变量进行赋值，最重要的赋值为 50，依次为 49,48，……，最后一个变脸赋值为 11。然后得到所有变量的集合，对于出现多次的变量将其值进行相加，得到了最终的特征排序加权集合 A5 取前 40 个变量展示如下图所示：

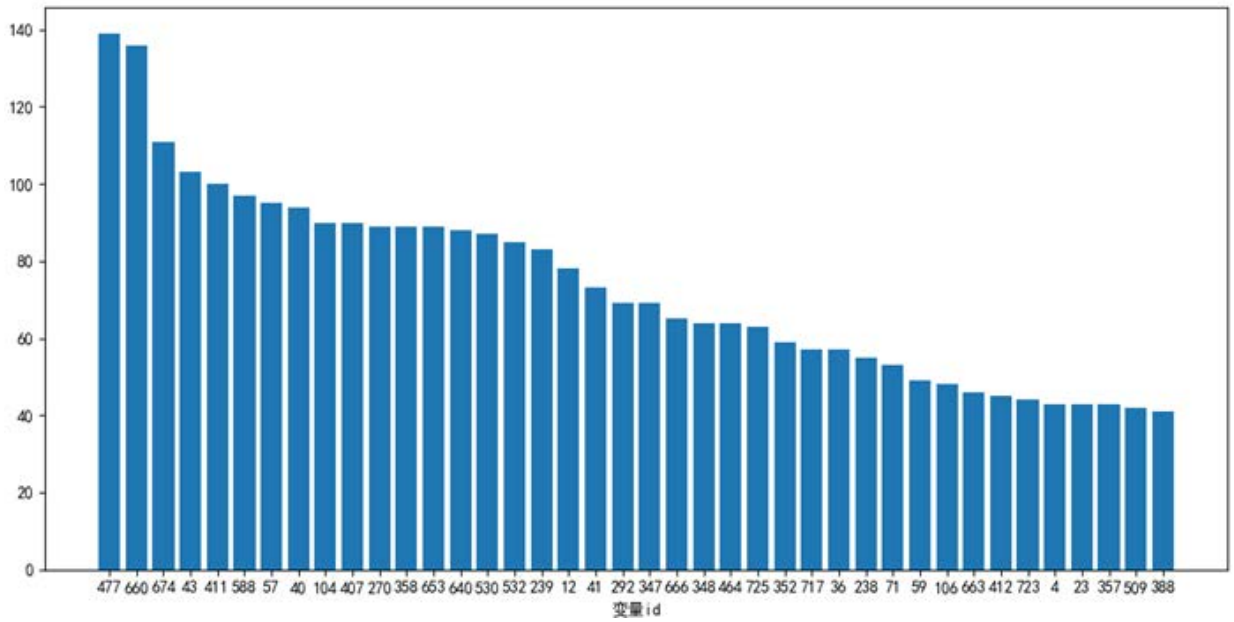


图 4.7 特征变量加权投票子集 A5 排序

从图 4.7 中可以看到出现多次的变量 a477 和变量 a660 排序最为靠前，也符合我们初步观察的预期。纵轴表示变量加权投票排序的综合值，既反映其出现的次数也反映了出现的先后顺序，保证特征选择的合理性。

4.4 变量独立性校验—高相关性滤波方法

具有高相关性的一对变量会增加数据集中的多重共线性，在分类与回归之前有必要变量子集独立性校验。变量筛选的思路是从高度相关的变量中删除贡献较小，仅保留贡献值最大的变量，认为它能代表所在高耦合变量组的全部信息。由于变量之间存在线性和非线性关系，仅仅描述线性相关的方法如Person相关分析并不适用。距离相关系数是为了克服Person相关系数的弱点而生的。在一些情况下，即便Person相关系数为0,我们也不能判断这个两个变量是独立的，因为两个变量有可能是非线性关系；但是距离相关系数如果是0，那么可以说两个变量是独立的。

对于两个随机变量$x \in R^p, y \in R^q$，记为$(x, y) = \{(x_i, y_i): i = 1, \cdots, n\}$作为给定的样本，$x$和$y$的距离相关系数(DCorr)的计算公式如下：

$$R^2(x,y) = \frac{v^2(x,y)}{\sqrt{v^2(x,x)v^2(y,y)}}$$

其中，

$$v^2(x,y)=\frac{1}{n^2}\sum_{i,j=1}^{n}A_{i,j}B_{i,j}$$

$$A_{i,j}=\|x_i-x_j\|_2-\frac{1}{n}\sum_{k=1}^{n}\|x_k-x_j\|_2-\frac{1}{n}\sum_{l=1}^{n}\|x_i-x_l\|_2+\frac{1}{n^2}\sum_{k,l=1}^{n}\|x_k-x_l\|_2$$

$$B_{i,j}=\|y_i-y_j\|_2-\frac{1}{n}\sum_{k=1}^{n}\|y_k-y_j\|_2-\frac{1}{n}\sum_{l=1}^{n}\|y_i-y_l\|_2+\frac{1}{n^2}\sum_{k,l=1}^{n}\|y_k-y_l\|_2$$

$$v^2(x,x)=\frac{1}{n}\sum_{i,j=1}^{n}A_{i,j}^2$$

$$v^2(y,y)=\frac{1}{n}\sum_{i,j=1}^{n}B_{i,j}^2$$

距离相关系数越大表示两个变量的相关性越强，相关性强弱判别关系可以见表 4.1。A5 变量自己前 40 个变量的距离相关系数热力图如图 4.8 所示。可以看到一些变量之间存在较强的相关关系如变量 a_{474} 和 a_{464}，根据我们筛选原则，应当保留排名靠前的变量 a_{474}。通常情况下去相关性之后的变量相关性应该在高相关性以下，所以将距离相关系数设置为 0.6。变量去相关性的的选取思路如下：

（1）选取变量 A5 子集的前 20 个变量计算相关距离系数，对于相关系数大于 0.6 的变量仅保留排名靠前的变量（排名依据图 4.7 加权投票排名）得到 *i* 个变量。

（2）从排名向后的变量中依次选取 20-*i* 个变量重新计算距离相关系数，同样对于相关系数大于 0.6 的变量仅保留排名靠前的变量，得到 *j* 个变量。

（3）重复步骤（1）和（2）知道选择出所有变量之前两项相关系数≤0.6。

经过筛选选择出的 20 个变量如下表所示（**问题一结果**）。

表 4.2 最终选择的 20 个变量（数据降维结果）

排序	变量序	分子描述符含义	排序	变量序号	分子描述符含义
1	477	maxHsOH	11	347	minHBa
2	660	MDEC-23	12	71	VCH-5
3	43	BCUTp-1h	13	352	minHBint5
4	411	minsOH	14	238	SHBint10
5	407	minsssN	15	717	TopoPSA
6	57	C1SP2	16	59	C3SP2
7	40	BCUTc-1l	17	663	MDEC-34
8	640	nHBAcc	18	412	mindO
9	532	maxssO	19	509	maxsssCH
10	41	BCUTc-1h	20	388	mindsCH

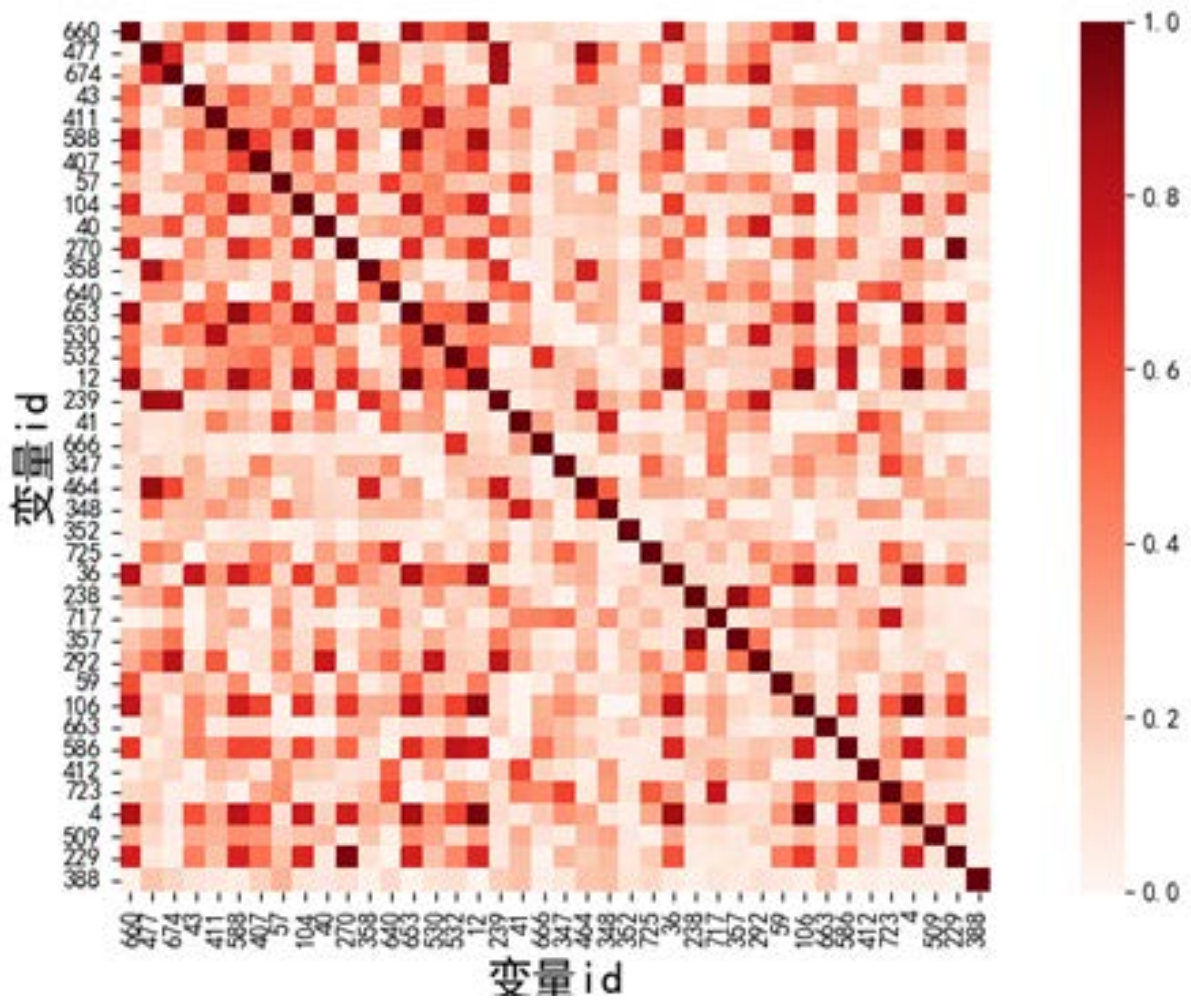


图 4.8 变量集 A5 前 40 个变量距离相关系数热力图

4.4 变量选择的合理性验证

（1）变量选择过程的方法和流程来看：

一方面在进行变量相关性计算时，共计采用了 4 种线性相关性计算和非线性相关性计算较好的方法，然后通过统一加权投票得到变量排序子集。这样可以过滤掉变量选择过程中存在偶然误差，保证了选择出的变量具有可靠性和稳定性。另一方面，在选择出的变量子集根据距离相关系数进行高相关系数滤波，保证选择出的变量具有一定的独立性，选择出的 20 个变量的相关关系热力图如图下图所示。可以看出仅仅有 4 对相关变量接近 0.6，但是实际小于 0.6。以上可以说明选取的 20 个变量的独立性较好。

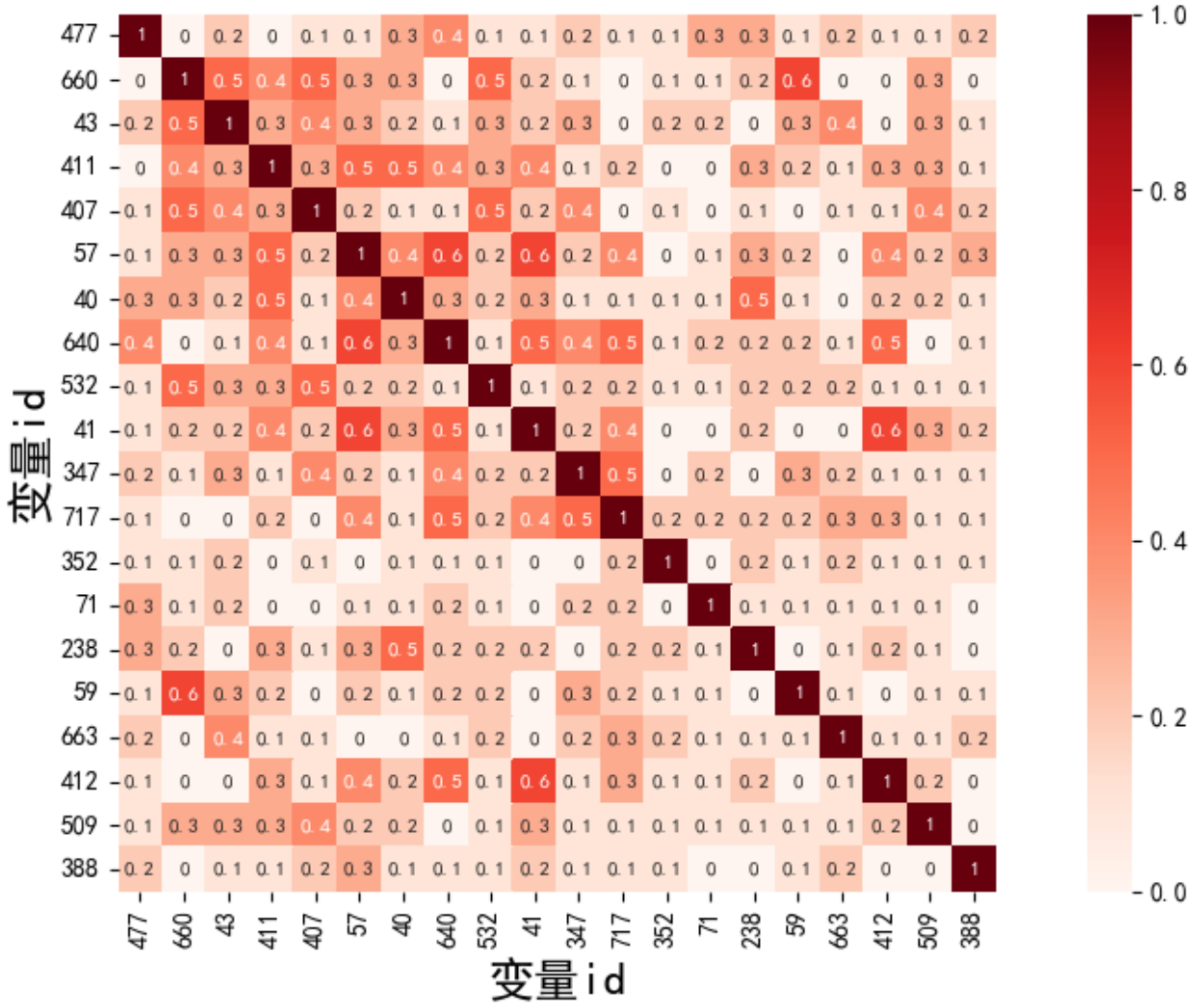


图 4.9 选择出 20 个变量距离相关系数热力图（注：为了便于观察，仅保留一位小数，等于 0.6 的值为四舍五入进位值）

（2）从选择出的结果来看：

根据选择出的 20 个分子描述符的类别进行分类统计得到下表。

表 4.3 变量分类统计表

分子描述符的类别	变量标号	数量
ElectrotopologicalStateAtomTypeDescriptor	477/411/407/532/347/352/238/412/509/388	10
BCUTDescriptor	43/40/41	3
CarbonTypesDescriptor	57/59	2
MDEDescriptor	660	1
PaDELHBondAcceptorCountDescriptor	640	1
TPSADescriptor	717	1
MDEDescriptor	663	1

从表格中变量数量占比来看，可以看到 ElectrotopologicalStateAtomType Descriptor 类别占据 10 个变量，但是同时该类别的变量在表格”分子描述符含义解释.xlsx”中占据超过 56%，因此选择出的变量也基本符合变量的占比。从变量类别分布来看，变量分属 7 个不同的类别，因此变量整体分布也较为合理。

5 问题二：pIC50 回归预测模型

5.1 问题二分析

问题二主要是基于问题一提取的 20 个变量构建构建化合物对 ERα 生物活性的定量预测模型。该问题本质是利用给定的数据集找到变量与因变量之间的函数映射关系$\boldsymbol{f}$：

$$\boldsymbol{Y}(pIC50) = \boldsymbol{f}(x), x \in \mathrm{A}$$

从问题一的变量相关性分析来看，变量与变量，变量与因变量之间存在较强的分线性关系。所以常用的基于线性回归的方法肯定在本问题不适用；由于变量较多，通过多项式拟合等方法来逼近非线性关系的方案显然十分困难。基于机器学习的数据分析和处理是近些年的热点，通过数据驱动的机器学习模型可以实现高度复杂大数据样本进行回归和预测分析。所以本题 pIC50 值作为因变量，20 个变量作为自变量，基于机器学习模型构建对化合物对 ERα 生物活性的定量预测模型。常用的回归模型有基于随机森林回归分析、BP 神经网络回归分析、支持向量机回归分析等经典模型。通过对查阅相关文献和对算法的对比我们初步选择基于直方图的梯度增强回归树（Histogram-based Gradient Boosting Regression Tree, HGBRT）。该算法以树为基本学习器的模型，也是目前数据回归统计领域中较为强大的以数据驱动的机器学习模型。该模型可以拟合数据之间的强非线性关系，实现数据的回归与预测分析。本文基于 python 强大的第三方库 sklearn 完成模型训练。为了验证模型的优势，将模型结果与随机森林、SVR、BP 神经网络方法进行横向对比实验验证分析。同时为了筛选出该模型最优超参数和最优变量，采用贝叶斯寻优、K 折交叉验证等方法对模型进行一步优化，建立最优变量组合下的最优回归模型。本问的思路流程图如下图所示：

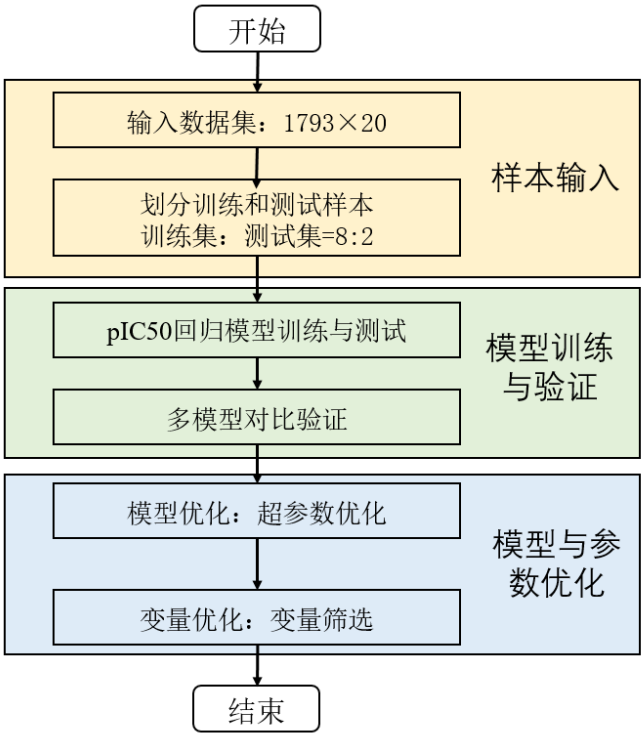


图 5.1 问题二建模思路分析

5.2 基于直方图的梯度增强回归树的 pIC50 预测模型构建

5.2.1 直方图梯度增强回归树(HGBRT)的原理简介

直方图梯度增强回归树也是 Boosting 学习方法中算法梯度提升树(Gradient Boosting Decison Tree, GBDT)的一员。GBRT 回归器是可加性模型，对其对给定输入 x_i 的预测形式 y_i 如下：

$$\hat{y}_i = F_M(x_i) = \sum_{m=1}^{M} h_m(x_i)$$

其中 h_m 为为弱学习器的估计值。通常梯度树增强使用固定大小的决策树回归器作为弱学习器，M 表示弱分类器的数量。

与其他 boosting 算法类似，GBRT 也是利用贪婪算法构建可得:

$$F_m(x) = F_{m-1}(x) + h_m(x),$$

其中新加入的树 h_m 是为了使损失 L_m 的总和最小化，给定之前预测集成预测 F_{m-1} 得到

$$h_m = \arg\min_h L_m = \arg\min_h \sum_{i=1}^{n} l(y_i, F_{m-1}(x_i) + h(x_i)),$$

其中，$l(y_i,F(x_i))$由损失函数定义。默认情况下，选择初始模型 F_0 作为损失最小化的常数:对于最小二乘损失，这是目标值的经验平均值。初始模型也可以通过初始参数指定。用一阶泰勒近似，损失函数 l 的值可以近似如下：

$$l(y_i, F_{m-1}(x_i) + h_m(x_i)) \approx l(y_i, F_{m-1}(x_i)) + h_m(x_i)\left[\frac{\partial l(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{m-1}}.$$

其中$\left[\frac{\partial l(y_i,F(x_i))}{\partial F(x_i)}\right]_{F=F_{m-1}}$表示损失对第二个参数的导数，数值上等于 $F_{m-1}(x)$。对于任意给定的 $F_{m-1}(x_i)$很容易计算，因为损失函数是可以微分的，将其表示为 gi。除去常数项可得

$$h_m \approx \arg\min_h \sum_{i=1}^{n} h(x_i) g_i$$

如果将 $h(x_i)$拟合预测一个与负梯度$-gi$ 成比例的值，则该值最小。因此，在每次迭代时，都拟合估计器 hm 来预测样本的负梯度。梯度在每次迭代时更新。这可以看作是泛函空间中的某种梯度下降。最终得到的提升树模型为

$$h_M(x) = \sum_{m=1}^{M} g_m(x)$$

基于直方图的梯度增强分类器由于其直方图特征的基准精度和相对于梯度增强回归器的计算代价而被使用。与梯度增强分类器不同的是，在基于直方图的梯度增强分类器中，通过特征直方图找到最优的分裂特征点。因此，直方图数据结构降低了计算复杂度。

与其他流行的机器学习模型相比，HGBRT 具有许多独特的优势，增加了其对机器学习系统的适用性。（1）HGBRT 可以开箱即用地处理缺失的数据，而其他机器学习模型则需要对训练数据进行额外的处理。常用的方法，如用平均值替换缺失的特征值，通常需要各方之间的协调，导致通信开销较大；（2）HGBRT 不需要大量的预处理操作，这节省了大

量的通信成本，并减少了由于需要额外的通信而导致的私有数据泄漏。经典机器学习模型的性能，例如线性模型、逻辑模型和支持向量机等，往往严重依赖于良好的预处理策略。标准的预处理方法，如归一化和缩放，同样需要各方之间的协调，特别是当各方的本地数据集是异构的。这也是本文选取 HGBRT 模型的重要依据。

5.2.2 生物活性 pIC50 预测模型建立、训练与测试

本文采用基于 HGBRT 机器学习模型用于构建变量对生物活性的定量预测模型。具体步骤流程如下：

（1）样本处理

数据样本 1973 个化合物样本，每个样本包含问题一提取的 20 维向量（分子描述变量）。由于 HGBRT 不需要对数据进行标准化和归一化处理，因此本过程省略。样本按照 8:2 的比例随机划分为训练数据集（1578 个）和测试集数据集（395 个）。

（2）模型训练设置

本文基于 python 的 sklearn 的机器学习开源工具完成训练。可以直接调到封装好的类 sklearn.ensemble.HistGradientBoostingRegressor

在训练过程中需要指定模型的超参数。HGBRT 模型的超参数如下表所示：

表 5.1 HGBRT 模型内置参数设置与解释

序号	内置属性	参数解释	默认值
1	Loss	损失函数	平方差损失
2	learning_rate	学习率	0.1
3	max_iter	最大迭代次数	100
4	max_leaf_nodes	树的最大叶子节点	1
5	max_depth	树的最大深度	None
6	min_samples_leaf	每片叶子的最小样本数	20
7	l2_regularization	L2 正则化参数	0
8	max_bins	用于未丢失值的最大容器数	255
9	categorical_features	类别特性	None
10	monotonic_cst	每个特征上执行的单调约束	None
11	warm_start	重复掉头	False
12	early_stopping	早期停止开关	auto
13	scoring	评分参数用于是否早期停止	Loss
14	validation_fraction	训练数据预留的比例	0.1
15	n_iter_no_change	用于确定何时“提前停止”。	10
16	tol	迭代误差	1e-7
17	verbose	可视化输出标志位	0
18	random_state	随机数审查功能	None

在这超参数配置中，“*loss*”，“max_depth”，“*max_leaf_nodes*”和“*min_samples_leaf*”是基于决策树作为基分类器的重要参数，也是需要优化的几个重要参数。为了和机器学习模型进行公平的对比，在首次训练中采用默认参数进行预训练。

训练过程采用统一采用 K 折交叉验证的方法，即将训练集随机分成 K 份，每次选取 K-1 份作为新的训练集剩余 1 份作为测试集。K-折交叉验证对提高模型精确度具有较大帮助。

（3）损失函数

损失函数也是训练过程中的目标函数，为了得到的模型更加准确，需要建立合适的目标函数。通常目标函数为模型预测值与真实值的差值，假设预测值为：$\hat{y}=\{\hat{y}_1,\hat{y}_2,\cdots,\hat{y}_n\}$；真实值为：$y=\{y_1,y_2,\cdots,y_n\}$，则损失函数为 $loss=\hat{y}-y$，为了避免出现负值，将损失函数进行平方得到损失函数：

$$loss=(\hat{y}-y)^2$$

目标函数则希望损失最小：

$$\min(\hat{y}-y)^2$$

当连续多次迭代前后的损失函数变化小于阈值则模型训练完成。

（3）模型评价与验证。

为了验证回归模型是否准确，需要在测试集上对预训练的模型进行验证。通常需要一定的技术指标来验证模型的准确性，常用的指标有均方根误差(MSE)、平均绝对误差(MAE)、均方根误差（RMSE）、决定系数 R^2。

1）均方根误差 MSE：

$$MSE=\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i-y_i)^2$$

2）平均绝对误差 MAE：

$$MAE=\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i-y_i|$$

3）均方根误差 RMSE：

$$RMSE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i-y_i)^2}$$

以上评价指标范围为[0,+∞)，当预测值与真实值完全吻合时等于 0，即完美模型；误差越大，该值越大。

4）决定系数 R^2

$$R^2=1-\frac{\sum_i(\hat{y}_i-y_i)^2}{\sum_i(\overline{y_i}-y_i)^2}$$

R^2 的取值范围为[0,1]：如果结果为 0，说明拟合效果很差；如果结果是 1，说明模型无误。一般来说 R^2 越大越好。

5.2.3 生物活性 pIC50 预测模型结果与对比

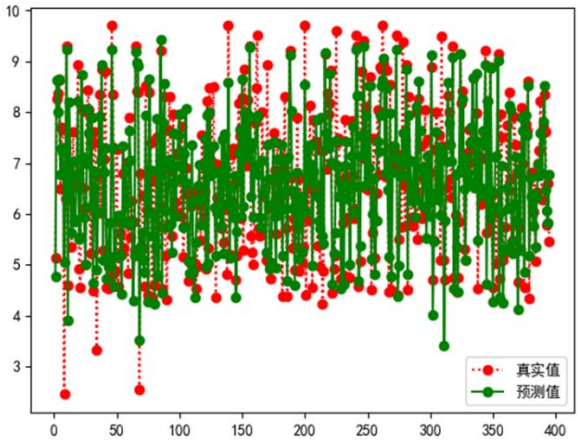
为了验证模型的有效性和优势，本文将其他多种模型的训练结果进行对比，为了保证公平我们均采用统一 sklearn 工具下函数拟合工具包内置模型，模型训练均采用默认参数。分别与多层感知机(MLP)、决策树（Decision tree）、随机森林（RF）、支持向量机回归（SVR）、梯度提升树（XGBRegressor）、Adaboost、Bagging 回归（BaggingRegressor）7 种常用的回归方法进行对比，得到结果如下表所示：（注：红颜色标记指标最好的模型算法）。所有方法均采用相同的数据样本进行预训练并在相同的测试集上进行测试，保证对比具有公平性。

表 5.2 多种回归模型的训练结果

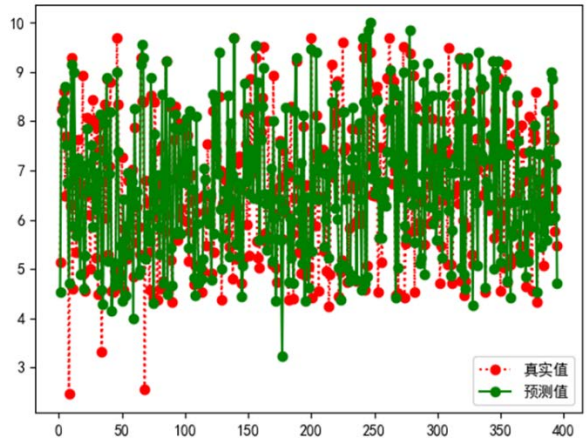
序号	回归模型	MSE↓	MAE↓	RMSE↓	R^2↑
1	MLP	0.545873	0.566467	0.738833	0.742522
2	Decisiontree	0.955678	0.702953	0.977588	0.549225
3	RF	0.495076	0.510539	0.703616	0.766482
4	SVR	1.486266	0.801904	1.219125	0.298957
5	XGBRegressor	0.916139	0.694157	0.957151	0.567875
6	Adaboost	0.895600	0.778986	0.946361	0.577563
7	BaggingRegressor	0.546729	0.549455	0.739411	0.7421183
8	本模型 HGBRT	0.463910	0.490858	0.681109	0.7811827

从表 5.2 可以看出本文采用 HGBRT 模型具有较小的 MSE、MAE、EMSE 说明模型的回归误差较好。HGBRT 模型也具有最好的 R^2 系数，说明模型的拟合程度较好。

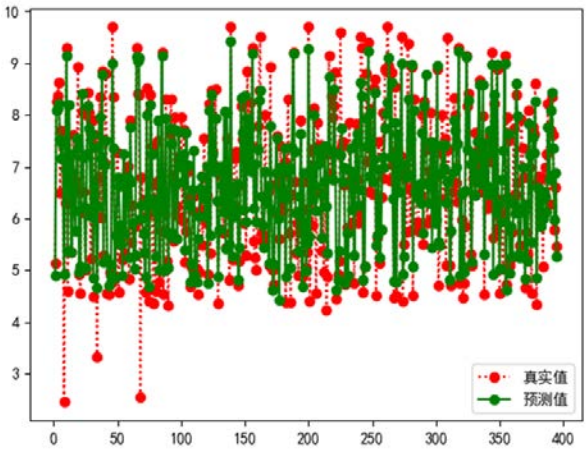
所有方法的在测试集上的预测值与真实值如下图所示



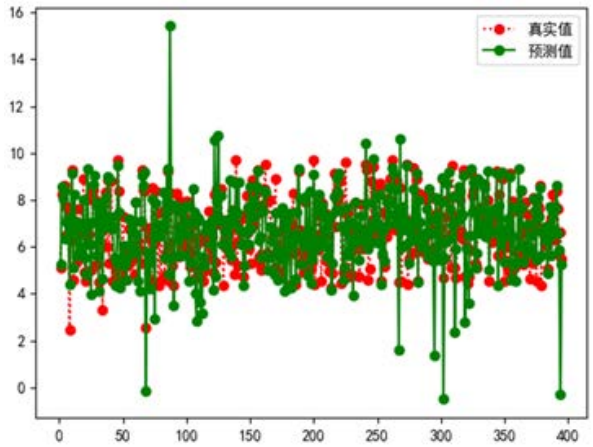
（1）MLP



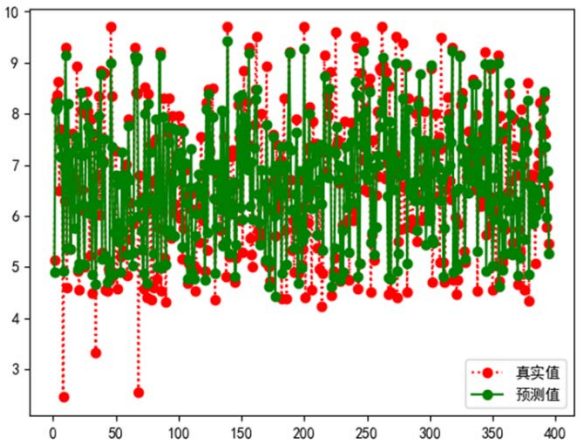
（2）Decisiontree



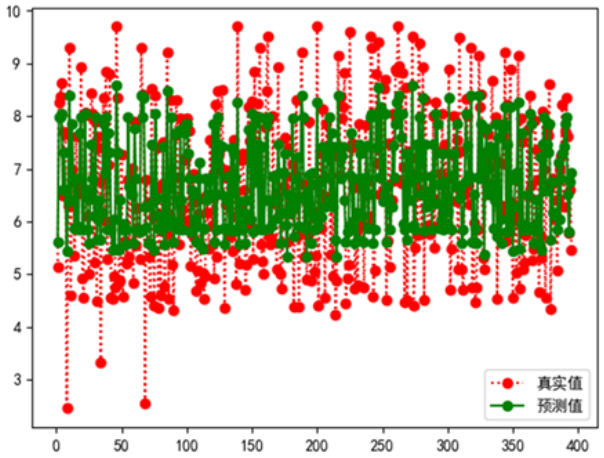
（3）RF



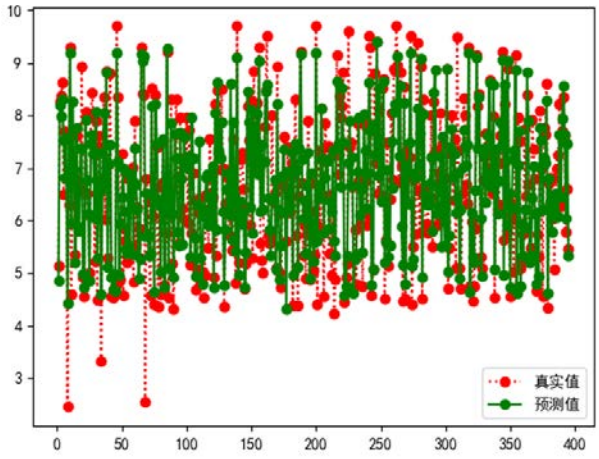
（4）SVR



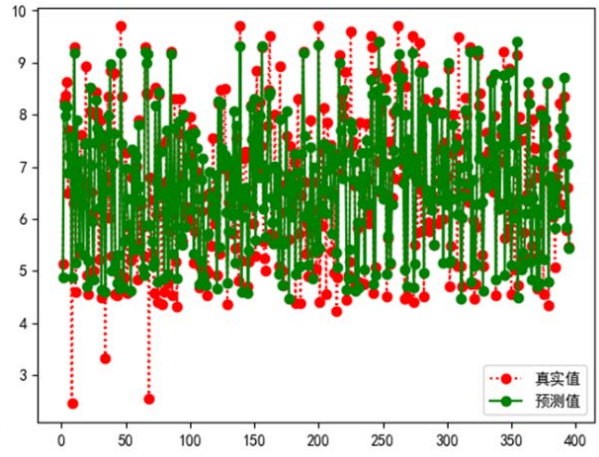
（5）XGBRegressor



（6）Adaboost



（7）BaggingRegressor



（8）本模型 HGBRT

图 5.2 在训练集的预测值和真实值对比图

从图中 5.2 中可以看出，相比于其他模型。本文采用的 HGBRT 回归模型，在绝大部份预测值都有较为准确，并且对于突然变化的极值没有预测到，也说明本模型训练没有过拟合。

5.3 HGBRT 回归预测模型优化与评价

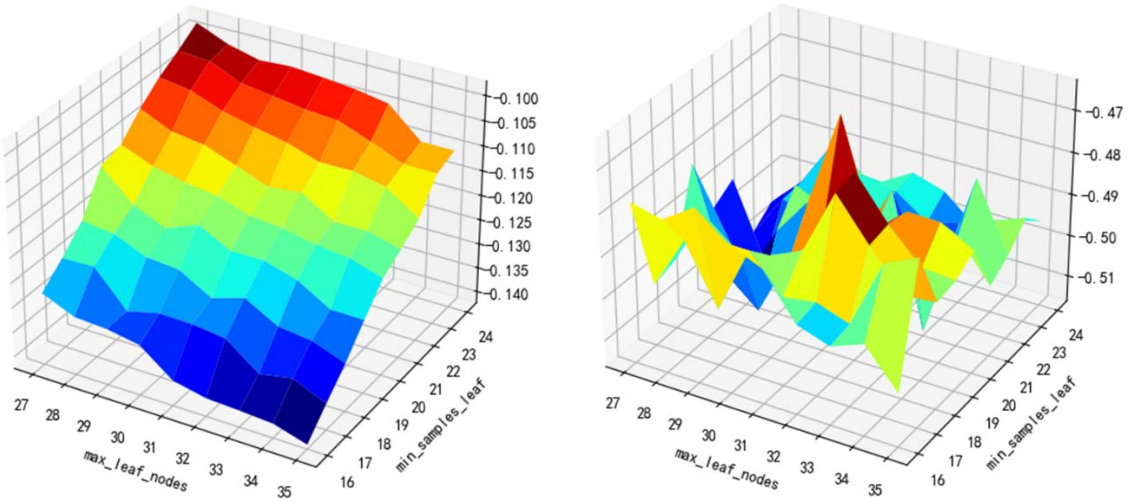
由于该模型是仅仅基于默认参数和 20 个筛选的变量进行训练，影响模型精度的主要因素有（1）模型预训练的超参数；（2）特征变量选取。因此对两个影响模型精度的因素进行优化。

（1）超参数优化

一些重要的参数需要进行优化达到模型的最优状态。HGBRT 是基于决策树作为基分类器，其中“max_depth”，“max_leaf_nodes”和“min_samples_leaf”三个参数是影响分类器性能主要的超参数，这些参数会影响模型中树的数量，叶子节点数量、每片叶子最小样本。“max_depth”默认值为 None 表示不设置深度，在模型拟合中会自动寻找到合适的深度，因此本文不再进行讨论。所以需要调整的超参数主要有“max_leaf_nodes”和“min_samples_leaf”。

超参数优化主要有网格搜索、随机搜索、贝叶斯优化等方法。由于本文需要优化的变

量仅有两个，不需要采用复杂的贝叶斯优化策略。首先，采用简单的网格搜索定位到大致范围，然后采用精细化调参对参数进行进一步优化。参数选择与误差可视化结果如下图。



（a）训练集　　（b）测试集

图 5.3 双变量超参数优化的可视化

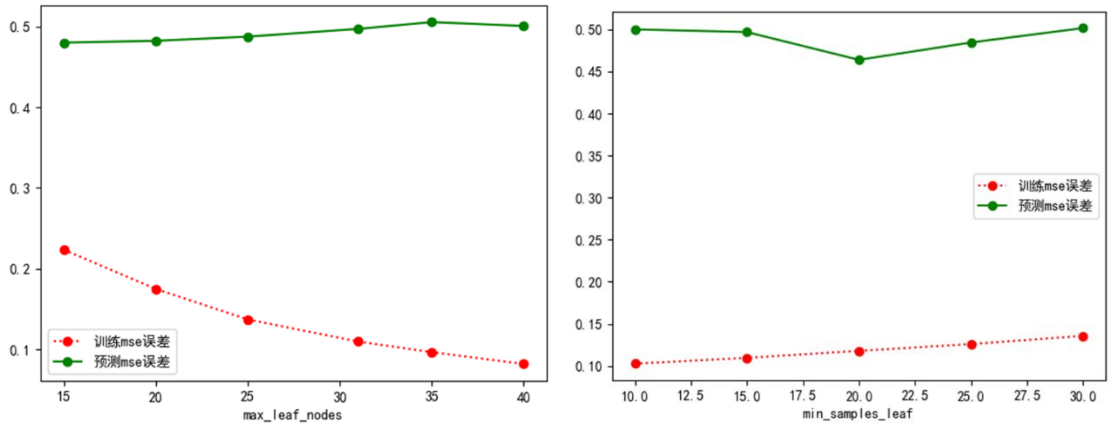


图 5.4 单变量参数优化结果的可视化

首先，从图中可以看出模型误差为一个量级，因此模型没有较多过拟合。然后，最大叶子节点数量和每片叶子最小样本数在训练集和测试集上的表现具有较大区别。最大叶子节点数量越多在测试集上的表现越好，而在测试集上先生后降。每片叶子最小样本数在在训练集上越小越好，在测试集上为 20 时，误差较小。

综合考虑，“max_leaf_nodes”取 40，“min_samples_leaf”取 20 作为最优的超参数。其他参数选取默认值，这样确定最优的 HGBRT 模型。

（2）变量优化筛选

在模型初步选择的 20 个，然而变量中可能存在相关关联等因素，导致对整个模型的误差扰动，考虑删去不重要的变量对模型选择合适参数。由于 HGBRT 和随机森林都是基于决策树作为基分类器，所有采用采用随机森林方法确定 20 个变量的重要程度排序如下图所示；

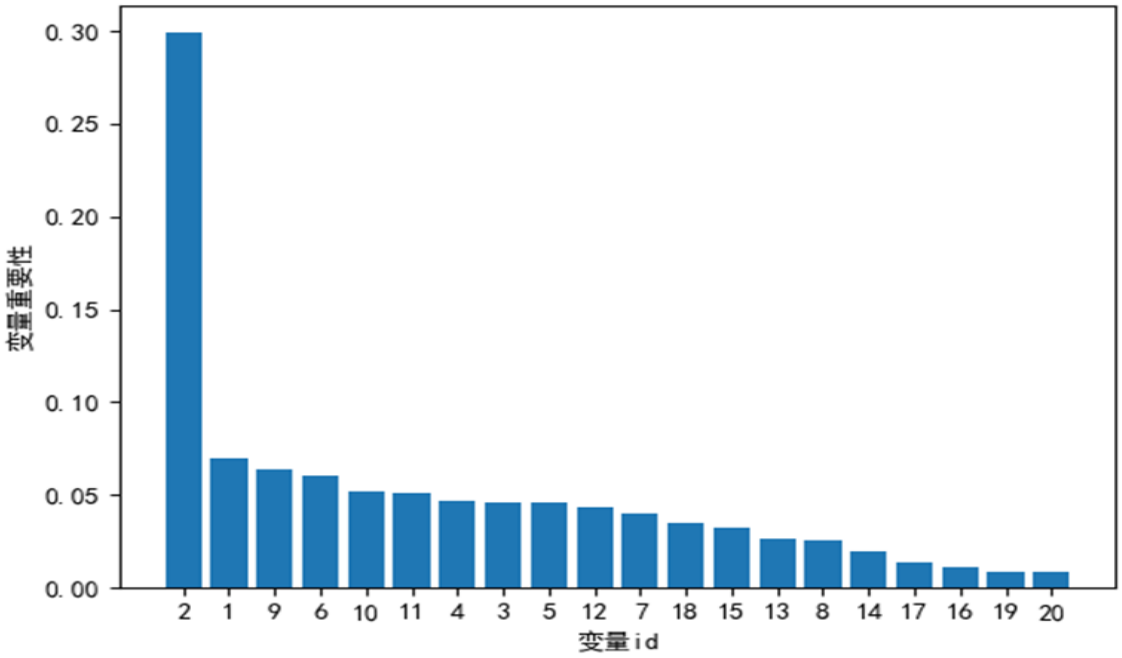


图 5.5 20 个变量的重要性排序（基于随机森林）

按照重要性从第到高依次删除变量，考虑删除变量后对模型精度的影响，在训练集和测试集上的结果如下图所示：

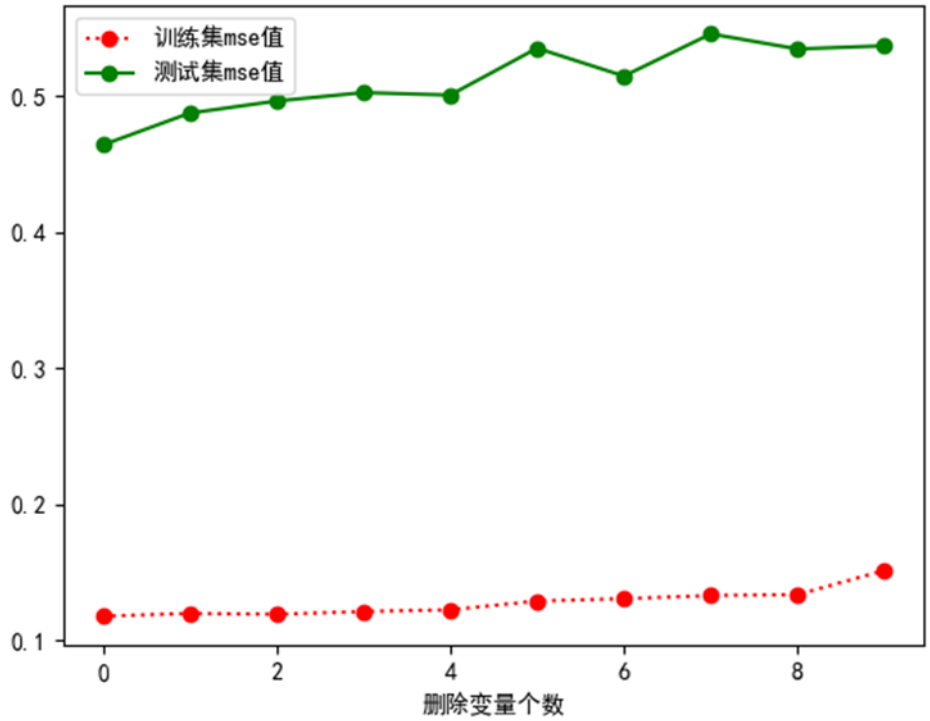


图 5.6 删除变量后模型预测精度变化曲线

从图 5.6 可以看出，模型删除 4 个变量后在训练集上的预测精度基本没有变化，在测试集上误差先增大在减小。当删除数量较多时发现模型误差在训练集和测试集都增加。这也从另一方面证明了问题一选择出的 20 个变量具有一定的合理性和代表性。所以，从出于减小变量数量出发，可以删除排序最后的 4 个变量，对模型精度几乎没有影响，即建议删除的变量为 C3SP2、MDEC-34、maxsssCH、mindsCH，剩余 16 个变量为表 4.2 中剩余变量。

（3）模型的整体评价

在变量选择和模型构建过程中都从多种角度出发考虑了模型的对比和验证过程，通过互相对比和验证考虑了模型建立的合理性，同时本文也考虑了模型的超参数优化和变量筛选。因此本文构建的 HGBRT 回归预测模型合理。

最终将选出的最优模型用于“ERα_activity.xlsx”teset 结果进行预测，结果见附录和附件

6 问题三：ADMET 分类预测模型

6.1 问题三分析

问题三主要是针对 1974 个化合物样本的 ADMET 数据构建 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型。模型的自变量需要从 729 个变量中重新筛选，模型的因变量是 0 和 1 二值向量，所以本问题为经典的二分类问题。数据集的正负样本分布如下图所示

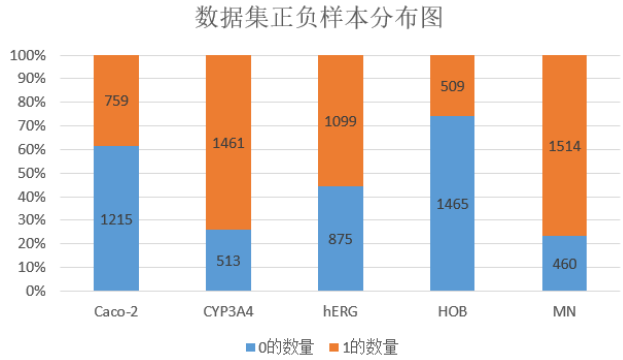


图 6.1 ADMET 数据集正负样本分布图

通过对数据集正负样本分析可以发现，CYP3A4、HOB 和 MN 三个变量的正负样本分布差异较大。通常情况下认为，正负样本比例小于 4:1 则认为样本分布不均衡，可能对模型误差分类产生较大的影响。通常处理样本分布不均衡的问题主要有以下三种方法：

（1）SMOTE(Synthetic Minority Over-sampling Technique)过采样小样本（扩充小类，产生新数据）：基于距离度量选择小类别下两个或者更多的相似样本，然后选择其中一个样本，并随机选择一定数量的邻居样本对选择的那个样本的一个属性增加噪声，每次处理一个属性。这样就构造了更多的新生数据，等效于对小类错分进行加权惩罚。

（2）欠采样大样本：将大类聚类成 N 个簇，然后使用每个簇的中心组成大类中的 N 个样本，加上小类中所有的样本进行训练。（优点是保留了大类在特征空间的分布特性，又降低了大类数据的数目，但是可能会存在仅描述大类样本局部特征的问题）

（3）对小类错分进行加权惩罚：对分类器的小类样本数据增加权值，降低大类样本的权值，从而使得分类器将重点集中在小类样本身上。在训练分类器时，若分类器将小类样本分错时额外增加分类器一个小类样本分错代价，这个额外的代价可以使得分类器更加“关心”小类样本。

针对分类问题，同样采用基于直方图的梯度增强分类树（Histogram-based Gradient Boosting Classifier Tree, HGBCT）实现对模型的构建。训练模型的主要变量依然利用问题一中的方法分别进行特征筛选和变量的独立性校验。整个问题三的思路分析流程图如下所示

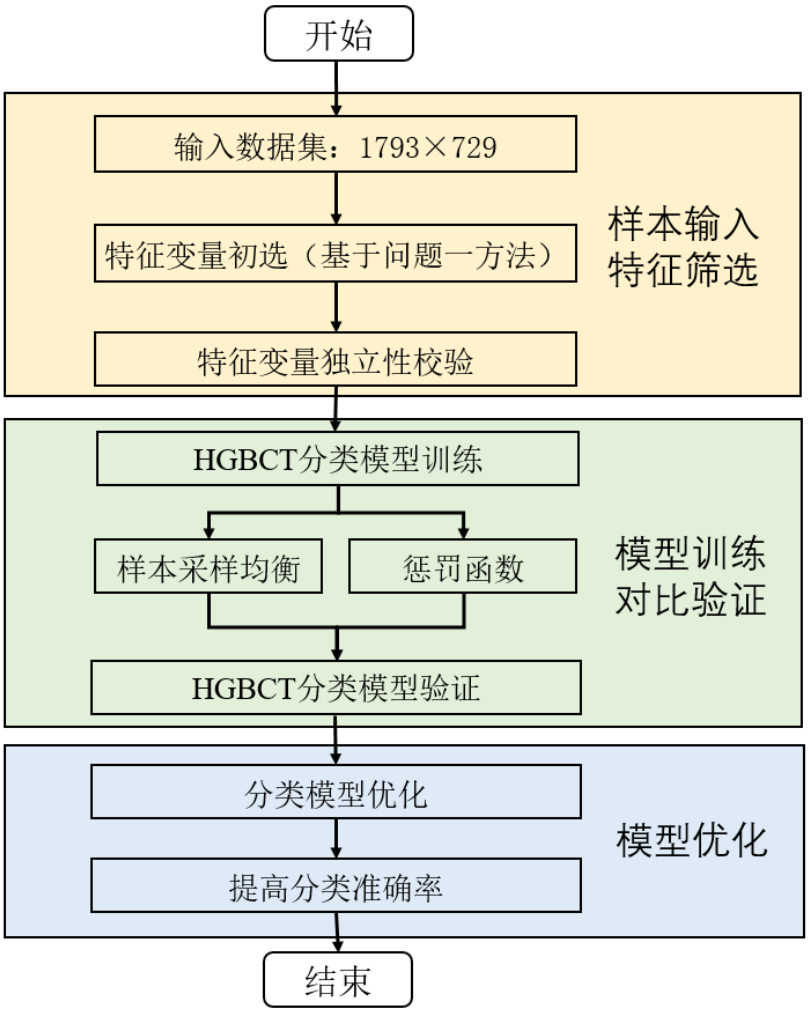


图 6.2 问题三模型思路分析

6.2 分类预测模型建立

6.2.1 基于直方图的梯度增强分类树（HGBCT）简介

HGBCT 的基本原理与 HGBRT 的原理基本相似，只是将回归问题转化为了分类问题。HGBCT 采用的基分类器是分类决策树，同时模型的损失函数也有所不同，交叉熵损失是分类问题常用的损失函数。

熵定义为信息的期望值，表示随机变量不确定性的度量。假设当前样本集合 D 中一共有 n 个样本，第 i 个样本为 x_i，那么 x_i 的信息定义为：

$$l(x_i) = -log_2 p(x_i)$$

其中，$p(x_i)$是选择该分类的概率。为了计算熵，需要计算所有类别所有可能值包含的信息期望数据，通过下面公式可以得到

$$Ent(D) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i)$$

Ent(*D*)的值越小，则 D 的不纯度就越低。对于一个二分类问题，模型最后需要预测的结果只有两种情况，对于每个类别我们的预测得到的概率为 *p* 和 1-*p*，此时的表达式

$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -[y_i \cdot log(p_i) + (1-y_i) \cdot log(1-p_i)]$$

其中，L 表示信息熵增益，这里为交叉熵损失。y_i 表示样本 *i* 的标签。正类为 1，负类为 0。这里 $p(x_i)$表示样本 *i* 预测为正类的概率。

6.2.2 分类问题评价标准

与回归问题不同，对于机器学习和数据挖掘中常用的评价指标有准确率（Accuracy）、精确率(Precision)、召回率(Recall)、F 值(F-Measure)。对于二分类问题存在 4 种分类可能，构成了混淆矩阵。

True Positive(真正，TP)：将正类预测为正类数

True Negative(真负，TN)：将负类预测为负类数

False Positive(假正，FP)：将负类预测为正类数误报 (Type I error)

False Negative(假负，FN)：将正类预测为负类数→漏报 (Type II error)

图 6.3 二分类问题的混淆矩阵

		预测类别		
		Yes	No	总计
实际类别	Yes	TP	FN	P（实际为Yes）
	No	FP	TN	N（实际为No）
	总计	P'（被分为Yes）	N'（被分为No）	P+N

1. 准确率（Accuracy）：

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

即正确预测的正反例数 /总数

2. 精确率、精度（Precision）

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

即正确预测的正例数 /预测正例总数

3.召回率（Recall）

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

即正确预测的正例数 /实际正例总数

4、F1-score

$$\text{F1score} = \frac{2Precision * recall}{precision + recall}$$

F 值是精确率和召回率的调和值，更接近于两个数较小的那个，所以精确率和召回率接近时，F 值最大。

6.2.3 基于 HGBCT 的 ADMET 分类模型建立、训练

（1）模型建立

由于影响化合物的 Caco-2、CYP3A4、hERG、HOB、MN 值的变量各不相同，需要针对每个 ADMET 性质建立单独的分类模型分别建立如下分类预测模型：

$$Y\{Y_1,Y_2,Y_3,Y_4,Y_5\}: \boldsymbol{f}(x_i \in X)$$

（2）变量筛选

在模型训练之前要选择合适变量。本问题采用问题一投票加权的方法初步提取排名前 20 的变量。经过独立性校验选择互相关系数不超过 0.6 的 20 个变量。

（3）模型数据集划分

划分数据集和训练集：样本 1973 个化合物样本，每个样本包含问题一提取的 20 维向量（分子描述变量）。由于 HGBRT 不需要对数据进行标准化和归一化处理，因此本过程省略。本题中正负样本比例接近 4:1 在可接受的范围之内。针对二分类中正负样本不均衡的问题，由于增加样本较小比例的方式可能会增加一些随机误差造成对模型的扰动；而减小大样本的数量可能造成对无法训练模型的整体特征。所以本文采用增加小样本分类错误的惩罚函数来解决该问题。样本按照 8:2 的比例为训练数据集（1578 个）和测试集数据集（395 个）。同时保证数据集和验证集中正负样本比例与整体的正负样本比例接近。

（4）模型损失函数（样本均衡处理方式）

模型损失函数：设置为交叉熵损失。增加对小样本分类错误的惩罚系数可以在 sklearn 中分类函数中的属性设置为 balanced 即（*class_weight='balanced'*），这样第 *i* 个类别的惩罚因子设置则则为 *class_weight* [i] * C(默认值 C=1)。同时在'balanced'模型下，使用 *y* 的值自动调整与输入数据中的类频率成反比的权重：

$$class_{wight[i]} = \frac{n\_samples}{n_{classes} * np.bincount(y)}$$

为了简单起见直接调节小类别的惩罚因子 C 也可以实现类似的功能。

（5）模型训练

模型采用 K-折交叉验证的方法，原理同问题二。

（6）模型评价

为了对比验证不同机器学习分类模型的准确率，对常见的 SVM、RF、MLP、Adaboost 和决策树分类器进行对比，本问题不讨论 0-1 分类中分类的好坏，即哪一类是正样本，哪一类是负样本。所以，采用准确率（Accuracy）作为整体的评价指标。

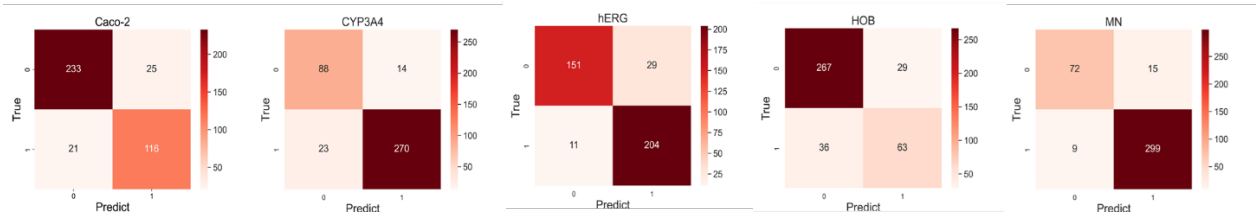
6.2.4 ADMET 分类结果对比

分类器训练中，通过调节不同的惩罚因子，得到不同分类器的分类准确率，其他参数均设置为默认参数。经过多次调试发现，惩罚因子 C 取值在 3~5 之间，分类结果较好。不同机器学习分类模型对 ADMET 的分类准确率在测试集的结果如表 6.2。

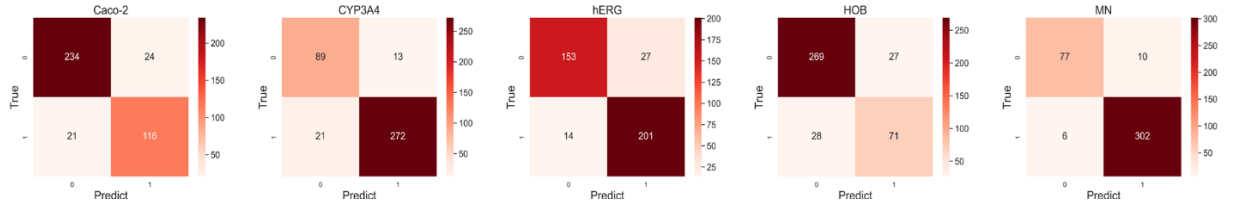
表 6.1 不同机器学习模型 ADMET 分类器分类准确率（Accuracy）（0~1）

序号	分类器	Caco-2	CYP3A4	hERG	HOB	MN	均值准确率
1	SVM	0.88	0.91	0.90	0.83	0.94	0.89
2	MLP	0.89	0.92	0.90	0.86	0.96	0.90
3	RF	0.90	0.93	0.93	0.88	0.98	0.92
4	Adaboost	0.86	0.91	0.89	0.85	0.95	0.89
5	DecisionTree	0.87	0.90	0.87	0.87	0.92	0.88
6	Our HGBCT	0.91	0.94	0.91	0.88	0.97	0.92

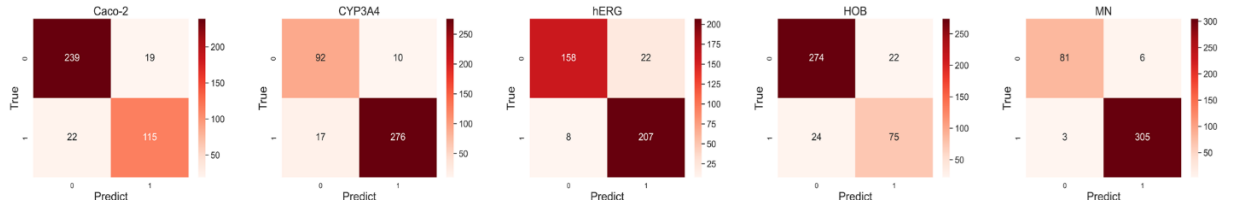
不同分类器对每个 ADMET 的分类结果混淆矩阵在测试的结果如图 6.4 所示。



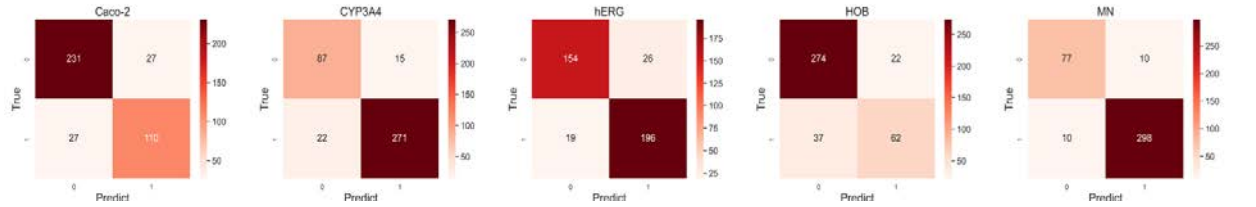
（1）SVM



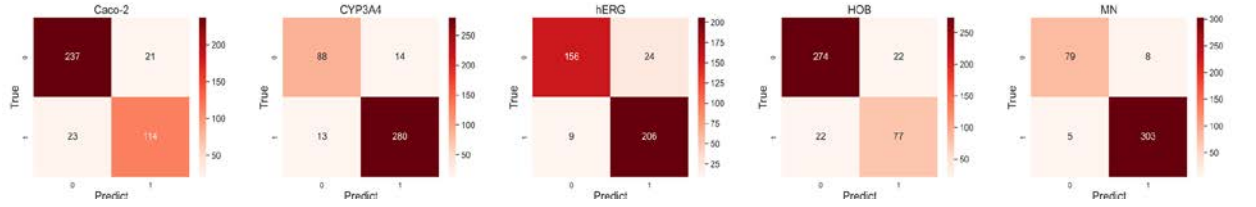
（2）MLP



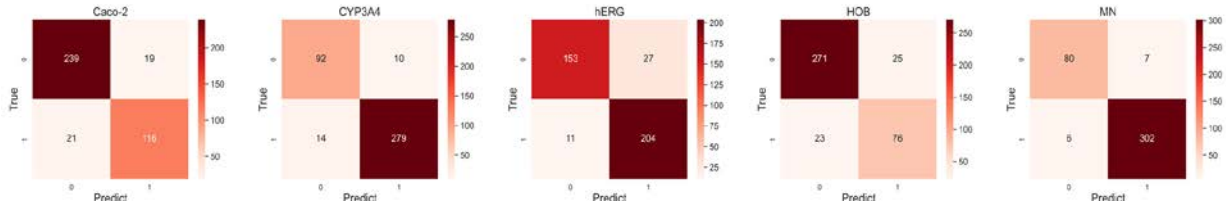
（3）RF



（4）Adaboost



（5）DecisionTree



（6）本文采用的 HGBCT 分类模型

图 6.4 多种机器学习分类模型对不用 ADMET 分类器的混淆矩阵

分类结果

从表 6.1 模型的分类准确率和图 6.4 的混淆矩阵来看，本文训练的 HGBCT 具有较高的分类准确率在对 Caco-2、CYP3A4、hERG 和 MN 的分类准确率中都达到了 0.91 以上，特别是对 MN 的分类准确率也达到了 0.97，对所有模型的平均预测准确也达到了 0.92。同时，随机森林分类器也具有较好的分类效果在对 hERG 和 MN 的分类中也略优于 HGBCT，并且对所有模型的平均分类准确率与 HGBCT 几乎一致。所以，随机森林分类器在本文题上也具有较好的性能。从深层次原理来分析，HGBCT 和随机森林都是基于 Bagging 方法的集成分类器，都是通过基于弱分类器（决策树）进行的多投票加权得到的结果。但是 HGBCT 对通过梯度提升树性能进行了优化，在处理大样本数据训练时具有更快的处理速度。

基于 HGBCT 分类器得到的“ADMET.xlsx”的表格中的分类预测数据填充至表格方法在附件中。

6.3 模型评价与分析

首选，在针对本文的分类问题处理中，考虑了 0-1 样本分布不均衡对模型分类产生影响的问题，主要采用增加小样本分类的惩罚系数来提升模型的分类精度。同时，采用多种分类模型对比，得出本文采用的 HGBCT 模型和随机森林这类基于决策树为基分类器进行投票的分类器具有较好的分类精度。

本文考虑了不同惩罚因子下对模型精度的影响进行分析，结果如下表 6.2 所示，对应的曲线变化趋势如图 6.5 所示。

表 6.2 不同惩罚因子下 HGBCT 模型分类准确率分布表

模型	1	2	3	4	5	6	7
Caco-2	0.867	0.862	0.879	0.887	0.903	0.902	0.894
CYP3A4	0.898	0.914	0.928	0.942	0.938	0.936	0.928
hERG	0.901	0.892	0.907	0.911	0.913	0.908	0.891
HOB	0.821	0.815	0.853	0.887	0.894	0.893	0.875
MN	0.954	0.956	0.963	0.972	0.971	0.966	0.964

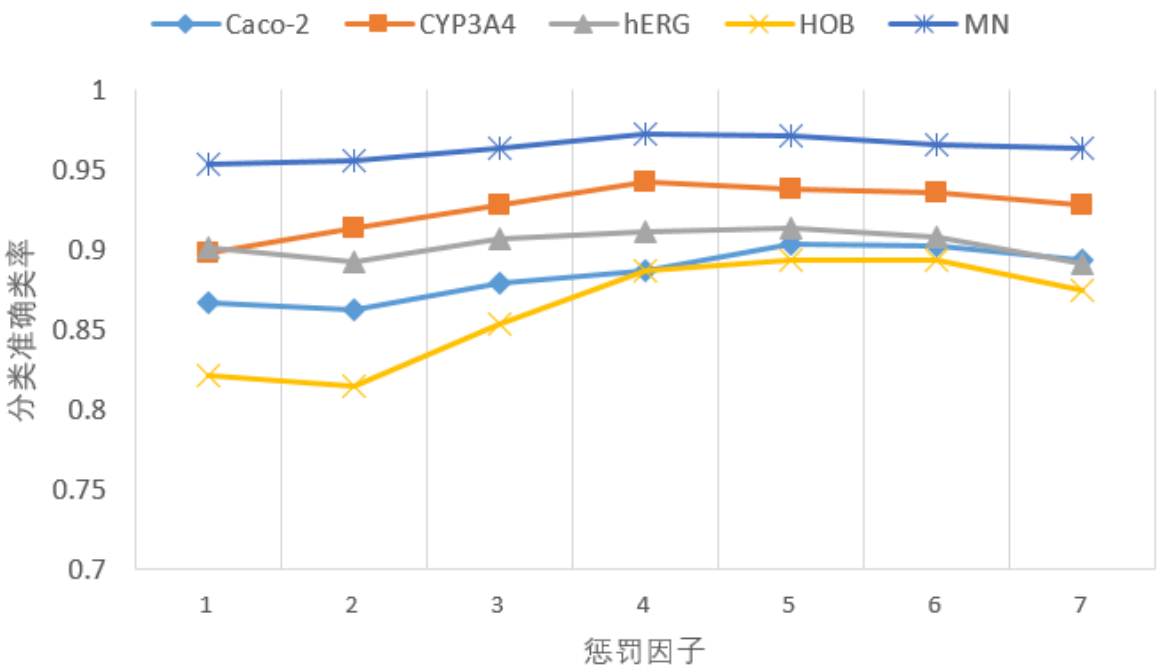


图 6.5 不同惩罚因子下 HGBCT 模型分类准确率变化曲线

综合表 6.2 和图 6.5 可以看出，惩罚因子对于模型的不同模型分类准确率各有不同，对于 HOB 分类模型，通过增加惩罚因子可以对分类准确率有较大的提升，分类准确率提高有 7%左右。而对 MN 分类模型而言，提升仅有 2%。造成这种情况的主要原因可能有：（1）变量提取合理性（2）提取变量与样本的相关关系。

综合来看，模型的惩罚因子 C 取值为 3~5 之间模型分类准确率最高。这个结论对模型训练超参数的选取具有重要意义。

7 问题四：最优变量优化建模

7.1 问题四分析

问题四需要找到影响化合物性质的分子描述符，并确定其取值范围使得化合物对 ERα 抑制具有更好的生物活性，同时具有更好的 ADMET 性质。

本问题可以看做是一个基于约束的多目标优化问题。本问题的难点在：（1）确定需要优化的份子描述符（变量），由于有些变量对 ERα 抑制有活性作用明显，而其对 ADMET 性质影响不明显。（2）变量之间可能存在共同的化学作用产生协同影响，导致在优化问题时产生联动效应。（3）有些变量取值为连续值，无法通过枚举搜所方法找到最优解，由于变量为高维空间，单纯的梯度下降方法求解难以保证求得的解为全局最优解。

针对以上问题，初步的建模思路为：首先确定优化变量。对 ERα 抑制具有更好的生物活性先关性较强的变量（问题一二已经解决），然后找到影响 ADMET 性质的变量（问题三已经解决），通过投票加权法找到能联合影响两者性质的公共变量作为需要初步优化的变量。对于具有单方面影响的变量，通过对相关样本的聚类采用优秀样本均值进行替代。然后，构建双目标优化模型。为了能找到全局最优解，提高搜索效率，采用目前最新的第三代非支配排序遗传算法(NSGA-III)，进行模型优化。找到满足题目样本对应的优化后的变量区间。本问题的思路流程图如下：

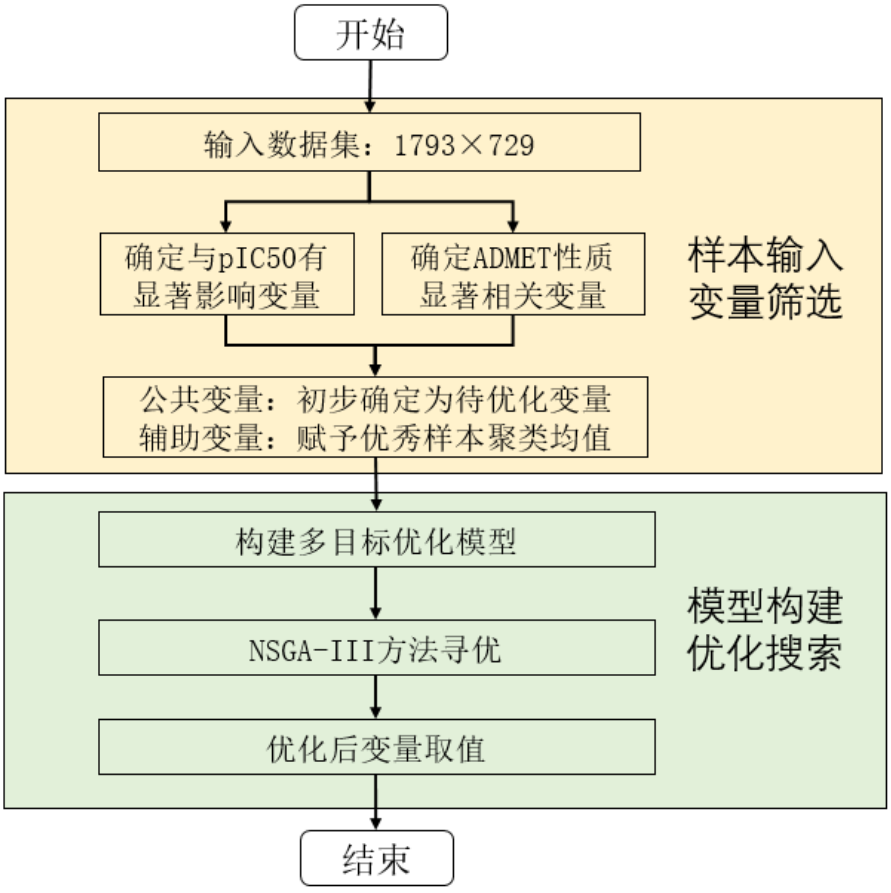


图 7.1 问题四思路分析流程图

7.2 确定待优化变量

对生物活性显著影响变量筛选已经在问题一中选择出，而对于 ADMET 性质具有显著影响的变量在问题三中分别选择出。为了筛选出合适的变量，本文采用加权投票的方式进行对变量进行排序。为了保证变量选择对 pIC50 和 ADMET 性质整体具有相同的影响，pIC50 变量的取值为[100,5]间隔为 5，ADMET 取值为[20,1]间隔为 1。具体过程如图 7.2 所示。

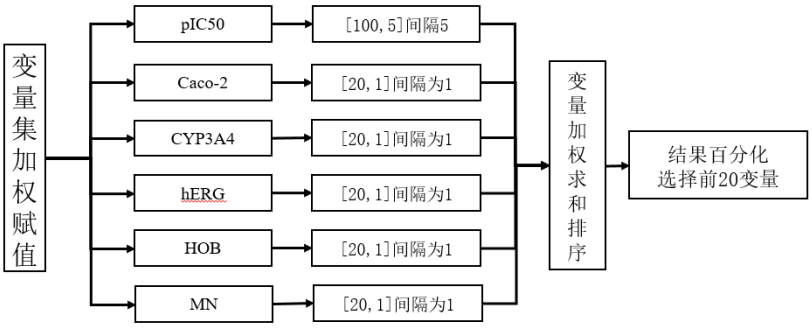


图 7.2 变量加权投票选择方法

变量加权投票选择结果与排序分别如表 7.1 和图 7.3 所示。变量集合记为 X_{20}。

表 7.1 最终选择的 20 个优化变量

排序	变量序	分子描述符含义	排序	变量序号	分子描述符含义
1	40	BCUTc-1l	11	611	ETA_Shape_P
2	347	minHBa	12	530	maxsOH
3	41	BCUTc-1h	13	677	MLFER_S
4	717	TopoPSA	14	678	MLFER_E
5	39	BCUTw-1h	15	505	maxssCH2
6	477	maxHsOH	16	43	BCUTp-1h
7	660	MDEC-23	17	411	minsOH
8	407	minsssN	18	532	maxssO
9	57	C1SP2	19	71	VCH-5
10	640	nHBAcc	20	352	minHBint5

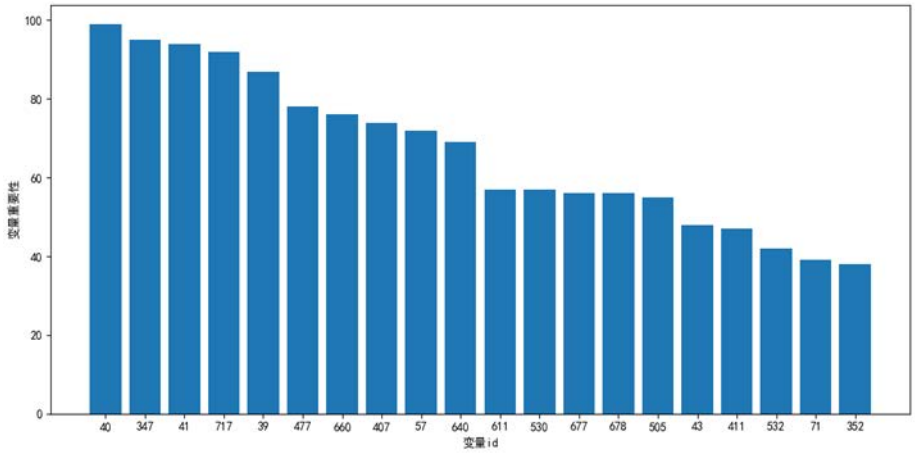


图 7.3 待优化变量投票加权排序图（百分制后）

7.3 基于 NSGA-III 双目标优化模型

NSGA-III 算法是遗传算法进行的优化，是典型的多目标进化算法，可以分为以五步：

（1）假设有一个规模为 *N* 的种群 A，用遗传算子（选择、重组、变异）对种群 A 进行操作，得到一个规模为 *N* 的种群 B，再将种群 A 和种群 B 混合，得到一个规模为 2*N* 的种群 C。

（2）对种群 C 进行非支配排序，得到非支配层级为 1、2、3、4……的诸层个体。把非支配层级为 1、2、3……的个体依次加入下一代子代的集合 D，直到集合 D 的规模大于 N。记下这个时候的非支配层级 L。然后从 L 层级里挑选出 K 个个体，使得 K 和之前所有层级个体之和等于 *N*

（3）函数标量化并对多目标函数进行标量化操作，以便下一步关联参考点。首先需要计算 *M* 个目标函数中每一个目标维度 *i* 上的最小值，得到第 *i* 个目标上对应的最小数值为 Z_i，此 Z_i 的集合即为理想点集合。然后利用 ASF 函数寻找极值点，遍历每个函数找到 ASF 数值最小的个体，依据这些函数值，算出对应坐标轴上的截距，记录为 a_i。得到 a_i 和 z_i 的具体数值后进行归一化运算。

$$ASF(X,W)=MAX_{i=1:m}\frac{f_i'(x)}{W_i}$$

$$f_i^n(x)=\frac{f_i'(x)}{a_i}$$

（4）关联个体参考点，构建参考点向量。然后针对每一个种群个体遍历所有向量，找到距离每个总群个体最近的参考点，记录参考点的信息和对应的最短距离。

（5）筛选子代删除参考点。经过非支配排序后，假设从第一个非支配层级到第 *FL* 层级的种群成员数目总和第一次超过种群规模 *N*，那么定义 *St*+1 为包含了 *FL* 中全部个体的集合，由于 St+1 的规模超过了预先设定的种群成员数目，因此需要进行相应的筛选。筛选的第一步是对每个参考点进行遍历，查看它被不包含 *FL* 的 St+1 引用的次数，并且寻找到被引用次数最少的参考点，也即被数量最少的种群个体所关联的参考点，将其被引用次数记录为 *pj*。然后对 *pj* 分情况讨论：1）假如这个参考点关联的种群个体数量为零，也即 *pj* 等于零，但在 *FL* 中有个体被关联到这个参考点向量，则从中寻找距离最小的点，并将其从 *FL* 中抽取，加入到被选择的下一代种群中，设置 *pj*=*pj*+1。2）如果在 *FL* 中没有个体被引用到该参考点，则删除该参考点向量，倘若 *pj*>0，则从中选择距离最近的参考点直到种群规模为 *N* 为止。

前面提到 NSGA-III 一个重要的方法是是非支配排序算法，对规模为 n 的中军进行分层的具体流程如下：

（1）设 i=1

（2）对于所有的 *j*=1,2,…,n 且 *j*≠*i*，比较个体 *xi* 和 *xj* 之间的支配与分支配关系。

（3）如果不能存在任何一个个体 *xj* 优于 *xi*，则 *xi* 标记为非支配个体；

（4）令 *i*=*i*+1,转到步骤（2），直到找到所有的非支配个体。

通过上述步骤得到的非支配个体集是种群的第一级非支配种群分层结束后，需要给每级指定一个虚拟适应度值，级别越小，说明其中的个体越优，赋予越高的虚拟适应值，反之级别越大，赋予越低的虚拟适应值。这样可以保证在复制操作中级别越小的非支配个体有更多的机会被选择进入下一代，使得算法以最快的速度收敛于最有区域。

7.3.2 多目标优化模型建立

依据问题四的要求和筛选出的 20 个变量建立如下多目标优化模型：

（1）目标函数：

$$\arg\max_{x\in X_{20}} F(x) := \lambda \times f_p(x \mid x \in X_{20}) + (1-\lambda) \times f_A(x \mid x \in X_{20})$$

（2）约束条件：

$$\text{s.t. } \{x_i \in [x_{ia}, x_{ib}] \mid x_i \in X_{20}\}$$

其中，λ 表示权重参数，表示对两种模型权重比值。f_p 为问题二得到的 Pic50 回归预测模型 HGBRT；f_A 为问题三的 ADMET 分类预测模型 HGBCT；$[x_{ia}, x_{ib}]$表示变量 x_i 的取值范围，默认值为该变量在数据集分布的最小值和最大值。

值得注意的是 PIC50 值为连续变化的变量，而 ADMET 为离散的类别，为了求解方便我们将 ADMET 的取值进行手动赋值。根据问题的叙述，要求 ADMET 性质最好，且只要有三项对人体有益，因此可以通过药物类别组合确定最优的药物组合：

1. 对人体 5 项最优的药物性质组合为只有 1 种可能为“10010”，赋值为 10.
2. 对人体 4 项最优的药物性质组合有 5 种可能，赋值为 8.
3 对人体 3 项最优的药物性质组合有 10 种可能，赋值为 5.
4. 对人体 2 项最优的药物性质组合有 10 种可能，赋值为 2.
5. 对人体 1 项最优的药物性质组合有 10 种可能，赋值为 1.
6. 对人体 0 项最优的药物性质组合有 1 种可能，赋值为 0.

对于类别赋值的的差异加入筛选最优药物性质的意愿。而 PIC50 的取值范围大约在 4-10 之间，因此两者数据尺度也合理，不会造成过多偏差。优化的目标如图 7.4 所示

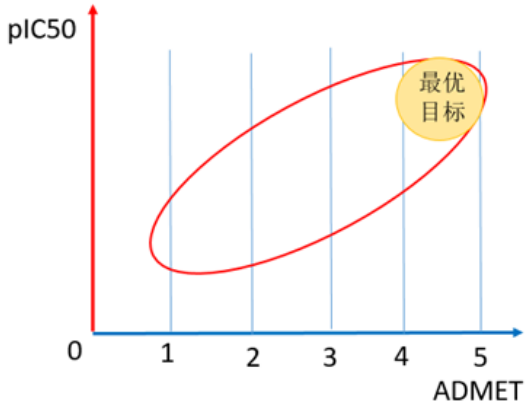


图 7.4 多目标优化模型

（3）NSGA-III 模型设置

1. 数据处理：

尽管 HGBRT 和 HGBCT 不需要进行数据归一化，但是为了消除各节点变量量纲在 NSGA 模型引起的误差需要对变量进行归一化处理，将数据归一化值[-1,1]。为了不影响数据的映射关系采用离差标准化形式对原始数据进行处理。x^*为标准化之后的数据，x 为原始数据，通过该式也便于还原真值。

$$x^* = \frac{x - max}{max - min}$$

2.染色体编码

本问题共计优化 20 个变量，染色体长度设置为 20。产生个体在（-1,1）之间随机产生

3.初始化种群

构建过程确定种群大小为 *PZ*，初始化种群为 20 行 PZ 列的矩阵。

4.适应度和选择操作

选择操作是从父代种群中根据计算的适应度值按照一定概率选择优良的个体组成新的种群。个体被选中的概率与适应度值有关，个体适应度越高被选中的概率越大，本文采用效率较高的轮盘赌法进行选择操作

5.交叉操作

交叉操作是从父代种群中随机选择两个个体，通过两个染色体交换组合，把父代的优良染色体遗传给子代进而产生新的优良个体。本文采用亮点交叉操作的方式。

6.变异操作

变异操作的作用是维持种群的多样性，从种群中随机选取一个个体，选择个体中的一个或几个染色体进行变异以产生更优秀的个体。本文选择多元非均匀变异算子完成变异操作。

7.遗传参数确定

遗传参数需要确定种群规模、交叉率、变异率、迭代次数。

种群规模分别设置为 100,200,300,400,500 进行尝试；

交叉概率设置为 0.1,0.2,0.3,0.4,0.5,0.6 进行尝试；

变异率设置为 0.01,0.03,0.05,0.07,0.08 进行尝试；

迭代次数设置为 300,400,500,700,1000 进行尝试。

当确定其中一个参数进行尝试时，其他参数均选择待选参数中值。

8.多目标优化权重设置

权重表示人工选择药物时对其哪种性质有更多的偏好，这里与设定为 $\lambda=0.5$，表示具有相同的偏好。

9.变量初始值设置

变量的初始值设置为该变量所有取值的均值，这样便于模型向两边进行搜索。变量的步长变化设置为变量取值范围的 1/100。

（4）模型求解

通过多次模拟实验确定 NSGA-III 最优遗传参数：种群规模 400，交叉概率：0.2，变异率：0.03，迭代次数：500（基本在 500 次后趋于稳定），权重参数为 0.5，取得的一组最优值如表 7.2 所示。

表 7.2 20 个变量取值

排序	变量序	描述符含义	取值	排序	序号	描述符义	取值
1	40	BCUTc-1l	-0.3617	11	611	ETA_Shape_P	0.1642
2	347	minHBa	6.1342	12	530	maxsOH	9.9176
3	41	BCUTc-1h	0.3674	13	677	MLFER_S	3.1974
4	717	TopoPSA	81.274	14	678	MLFER_E	3.8743
5	39	BCUTw-1h	31.9724	15	505	maxssCH2	0.7426
6	477	maxHsOH	0.5002	16	43	BCUTp-1h	14.2842
7	660	MDEC-23	28.7626	17	411	minsOH	8.1263
8	407	minsssN	2.4534	18	532	maxssO	6.0574
9	57	C1SP2	3.0135	19	71	VCH-5	0.0573
10	640	nHBAcc	6.0247	20	352	minHBint5	1.1278

7.4 多目标优化模型验证分析

经过带入表 7.2 的 20 个变量进行校验，所有的变量均在取值范围之内。在该参数下 HGBRT 模型得到的 PIC50 活性值为 10.098，活性值较高。ADMET 性质为“10011”，取值为 8。为具备 4 项最优属性，说明该模型具有较好筛选效果。由于本模型是求得的最大值优化问题，可以通过调节不同设置参数的范围，或者多次迭代来的得到最值范围。对于权重参数 λ，可以通过调节不同的权重来获得对 PIC50 和 ADMET 性质的偏好。通过设置不同 λ 取值得到函数最大值变化曲线如下图所示。

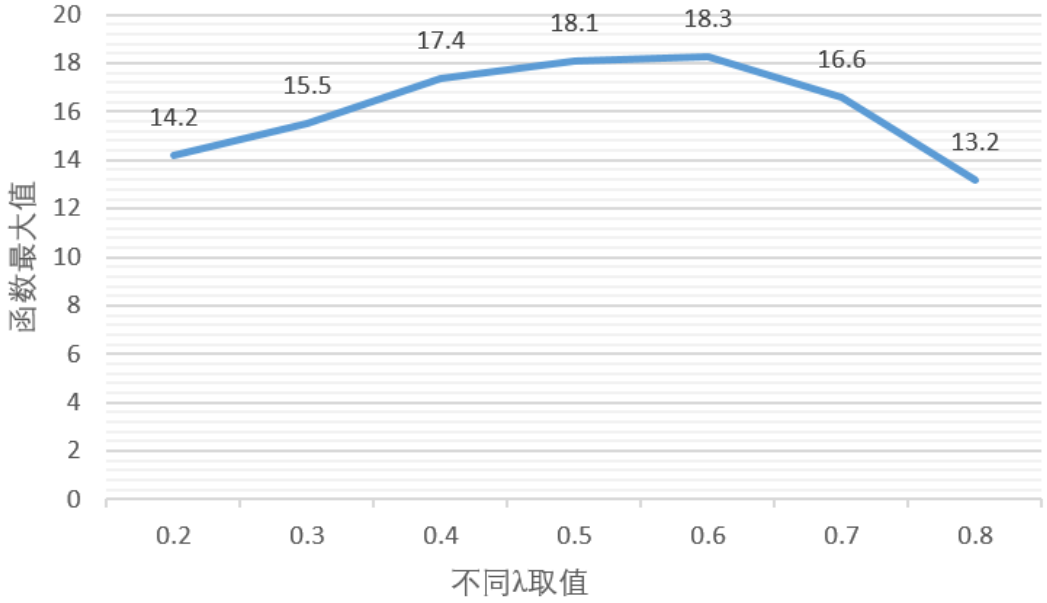


图 7.5 不同 λ 取值下模型最大值分布曲线

由图 7.5 可知权重在取不同值时，函数最值也不断变化，当权重取值在 0.5~0.6 之间时可能得到模型最优的取值范围。该变化曲线也从侧面证明了该模型具有一定的灵敏性，同时基于 NSGA-III 方法搜索方法也是可以实现最优变量的寻找。

8 模型评价与改进

8.1 模型优点

（1）充分考虑了变量之间的线性和非线性相关关系，采用多种方法投票加权方式获得最右变量，经过高相关滤波，保证了变量选取的合理性和独立性

（2）算法本文采用基于 HGBRT 和 HGBCT 分别构建了回归预测模型和分类预测模型，该模型基于决策树作为基分类器，不要对数据进行数据归一化处理。同时，该模型经过了科学家的多次优化，在兼顾模型精度的同时充分考虑了模型的训练效率，对处理大型数据样本特别是样本数量几万，几十万的海量数据与经典的机器学习模型具有明显优势。

（3）算法采用的 NSGA-III 算法，该算法在 NSGA 算法多次迭代而来，不仅具备多目标启发式寻优的优秀能力，特别是在对多变量寻优时具有更优秀的能力

8.2 模型缺点

（1）分析问题时，实际上变量之间可能存在复杂的物理化学相关性，为了简化模型本文假设其相互独立互不影响，这可能与实际不符。

（2）在选择变量时，仅仅选择了 20 个主要变量，这是人为规定的数量，没有对排序位于 20 之后的变量影响进行讨论

8.3 模型的推广与改进

1）本文提出的方法和模型思路适合大部分大样本和海量数据处理的数据挖掘问题，例如，生物化学以及材料科学中的变量筛选，以及其他数据挖掘的回归、预测和优化问题。

2）模型可以在考虑变量耦合性的复杂问题上进行一些探索和优化，以便处理更复杂的数据挖掘问题。

9 参考文献

[1] Hossain S.M.M., Deb K. (2021) Plant Leaf Disease Recognition Using Histogram Based Gradient Boosting Classifier. In: Vasant P., Zelinka I., Weber GW. (eds) Intelligent Computing and Optimization. ICO 2020. Advances in Intelligent Systems and Computing, vol 1324. Springer, Cham. https://doi.org/10.1007/978-3-030-68154-8_47

[2] sklearn: https://scikit-learn.org/

[3] 多变量筛选法：https://blog.csdn.net/fjsd155/article/details/93754257

[4]HGBRT 模型介绍：
https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingRegressor.html

[5] NSGA-III 方法与代码：https://deap.readthedocs.io/en/master/examples/nsga3.html

Deb, K., & Jain, H. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box

Constraints. IEEE Transactions on Evolutionary Computation, 18(4), 577-601. doi:10.1109/TEVC.2013.2281535.

10 附录

附件说明：

附录中包含文件

1）“ERα_activity.xlsx”表格包含问题二 test 预测的结果，

2）“ADMET_e.xlsx”表格包含问题三 test 预测的结果

3） 本文代码放在附件中，部分核心代码放在附录中。

完成本文需要的代码，基于 python3.6 以上版本完成，依赖的第三方库由 import 导入，如果没有安装需要手动安装。

表 10.1 问题二预测结果

IC50_nM	pIC50
33.275	7.478
55.464	7.256
88.944	7.051
80.488	7.094
14.242	7.846
160.551	6.794
65.022	7.187
56.781	7.246
121.293	6.916
14.744	7.831
24.414	7.612
12.176	7.915
5.480	8.261
16.960	7.771
26.337	7.579
17.328	7.761
9.238	8.034
325.107	6.488
153.278	6.815
11.280	7.948
1039.717	5.983
104.633	6.980
329.766	6.482
315.128	6.502
817.421	6.088

527.110	6.278
94.898	7.023
234.518	6.630
836.290	6.078
1048.418	5.979
4488.122	5.348
5318.970	5.274
3442.069	5.463
7181.087	5.144
15816.873	4.801
316.044	6.500
298.090	6.526
1146.972	5.940
533.047	6.273
339.109	6.470
313.006	6.504
313.006	6.504
339.109	6.470
213.512	6.671
313.006	6.504
24.118	7.618
60.259	7.220
57.830	7.238
82.260	7.085
1.480	8.830

表 10.2 问题三预测结果

Caco-2	**CYP3A4**	**hERG**	**HOB**	**MN**
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	0	0	1
0	1	0	0	1
0	1	1	0	0
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1

0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	0	0	1
0	0	0	0	1
0	1	1	0	1
0	1	1	0	1
1	1	1	0	0
1	1	1	0	0
1	0	1	0	0
1	1	1	0	0
0	1	1	0	0
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	1	1
0	1	1	1	1
1	1	1	1	1
0	1	1	1	1
0	1	1	0	1
0	1	0	0	1
0	1	0	0	1
0	1	1	0	0
0	1	0	0	1
0	1	0	0	1
0	1	0	0	1
0	1	0	0	1
0	1	0	0	1
0	1	0	0	1
0	1	0	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	0

部分核心代码:

1）问题 1 代码

```
if __name__ == "__main__":
    # read files
dill.load_session('pro3_1.pkl')
    # 变量标准化
    scaler_train = StandardScaler()
    scaler_train.fit(Mol_train_final_array)
    all_data = scaler_train.transform(Mol_train_final_array)
    all_label = ERa_train_final_array[:, 1]
    # 投票多个相关性
    [feature_array_weight, score_array] = seleimportpambyvote(all_data, all_label)
    print(feature_array_weight)
    # 各个变量之间的解耦
[feature_array_weight_last, score_array_last] = decoup(all_data, feature_array_weight, score_array)

def seleimportpambyvote(all_data, all_label):  #投票法选
    # corr calc
    feature_list_1, feature_1 = LassoImportance(all_data, all_label)
    feature_list_2, feature_2 = RFImportance(all_data, all_label)
    feature_list_3, feature_3 = MIRmportance(all_data, all_label)
    feature_list_4, feature_4 = coorImportence(all_data, all_label)
    plotbar(feature_list_1, feature_1)
    plotbar(feature_list_2, feature_2)
    plotbar(feature_list_3, feature_3)
    plotbar(feature_list_3, feature_3)

    # calc the score
    score_list = []
    for i in range(730):
        score_list.append(0)  # initialize the score list
    for i in range(40):
        score_list[feature_list_1[i]] += 50 - i
        score_list[feature_list_2[i]] += 50 - i
        score_list[feature_list_3[i]] += 50 - i
        score_list[feature_list_4[i]] += 50 - i

    score_array = np.array(score_list)
    feature_array_weight = np.argsort(-score_array)[0:40]
    # plotbar(feature_array_weight, score_array[feature_array_weight])
    return feature_array_weight, score_array[feature_array_weight]
# 相关性解耦
def decoup(all_data, feature_array_weight, score_array):
```

```
        print('now coorImportence')
        all_data_pd = pd.DataFrame(all_data[:, feature_array_weight], dtype=np.float)
        corr_array = all_data_pd.corr('spearman')    # 'pearson', 'kendall', 'spearman'
        corr_array = abs(corr_array)

        plotheatmap(feature_array_weight, corr_array)

        inddele = []
        for i in range(40):
            for j in range(i+1, 40):
                if abs(corr_array.at[i, j]) > 0.95:    # 0.65
                    if score_array[i] > score_array[j] and not(j in inddele):
                        inddele.append(j)
                    elif score_array[i] <= score_array[j] and not(i in inddele):
                        inddele.append(i)

        feature_array_weight_cor = np.delete(feature_array_weight, inddele, axis=0)
        score_array_cor = np.delete(score_array, inddele, axis=0)

        #   对删除后的 sore 选前 20 个
        score_array_cor_ret = score_array_cor[0:20]
        feature_array_weight_ret = feature_array_weight_cor[0:20]

        # 画最后的输出的 20 个变量的图
        datacorr = pd.DataFrame(all_data[:, feature_array_weight_ret], dtype=np.float)
        corr_array_plot = datacorr.corr('spearman')    # 'pearson', 'kendall', 'spearman'
        plotheatmap(feature_array_weight_ret, abs(corr_array_plot).round(1))

    return feature_array_weight_ret, score_array_cor_ret
```

2）问题 2 代码

```
if __name__ == "__main__":
        dill.load_session('pro1_1.pkl')
        rcParams['axes.unicode_minus'] = False
        mpl.rcParams['font.sans-serif'] = ['SimHei']
        plt.rcParams['font.sans-serif'] = ['SimHei']
        plt.rcParams['axes.unicode_minus'] = False
        all_data = Mol_train_final_array[:, feature_array_weight_last]
        all_labels = ERa_train_final_array[:, 1]
        scaler_train = StandardScaler()
        scaler_train.fit(all_data)
        all_data = scaler_train.transform(all_data)
        # 划分训练集和验证集
        xls_test_x_array = Mol_test_final_array[:, feature_array_weight_last]
```

```
        xls_test_x = scaler_train.transform(xls_test_x_array)
        train_data, test_data, train_labels, test_labels = train_test_split(all_data, all_labels, test_size=0.2,
    random_state=1)
        train_MLP(train_data, train_labels, test_data, test_labels, scaler_train)
        train_DecisionTreeRegressor(train_data, train_labels, test_data, test_labels, scaler_train)
        train_RF(train_data, train_labels, test_data, test_labels, scaler_train)
        train_SVR(train_data, train_labels, test_data, test_labels, scaler_train)
        train_XGBRegressor(train_data, train_labels, test_data, test_labels, scaler_train)
        train_LinearRegression(train_data, train_labels, test_data, test_labels, scaler_train)
        train_AdaBoostRegressor(train_data, train_labels, test_data, test_labels, scaler_train)
        train_ExtraTreesRegressor(train_data, train_labels, test_data, test_labels, scaler_train)
        train_HistGradientBoostingRegressor(train_data, train_labels, test_data, test_labels, scaler_train)
        train_BaggingRegressor(train_data, train_labels, test_data, test_labels, scaler_train)
        train_LGBMRegressor(train_data, train_labels, test_data, test_labels, scaler_train)
        train_BestRegressor(train_data, train_labels, test_data, test_labels, scaler_train)
        plot_BestRegressor(train_data, train_labels, test_data, test_labels, scaler_train)
        pamselect_Regressor(train_data, train_labels, test_data, test_labels, scaler_train)
    xls_test_compute(train_data, train_labels, test_data, test_labels, xls_test_x, scaler_train)

    def train_BestRegressor(train_x, train_y, test_x, test_y, scaler=None):
        print('now training BestHistGradientBoostingRegressor...')
        # param_grid = [{'loss': ['squared_error'],
        #                'max_leaf_nodes': [i for i in range(25, 45)],
        #                'learning_rate': [0.1],
        #                'min_samples_leaf': [i for i in range(10, 25)], }]
        param_grid = [{'max_leaf_nodes': [29, 30, 31, 32, 33],
                       'min_samples_leaf': [18, 19, 20, 21, 22], }]
        classifier = HistGradientBoostingRegressor()
        grid_search = GridSearchCV(classifier, param_grid, cv=5,
                                   scoring='neg_mean_squared_error')
        grid_search.fit(train_x, train_y)
        print(grid_search.best_params_)
        classifier = HistGradientBoostingRegressor(learning_rate=0.1, loss='squared_error', max_leaf_nodes=31,
                                                   min_samples_leaf=20)

        classifier.fit(train_x, train_y)
        test_pred = classifier.predict(test_x)
        mse = mean_squared_error(test_y, test_pred)
        mae = mean_absolute_error(test_y, test_pred)
        rmse = mse ** 0.5
        r2 = r2_score(test_y, test_pred)
```

```
        print('MSE:', mse, 'MAE:', mae, 'RMSE:', rmse, 'R2:', r2)
        plt.figure()
        x = np.arange(1, len(test_y) + 1)
        plt.plot(x, test_y, linestyle=":", marker='o', color='r', label='真实值')
        plt.plot(x, test_pred, linestyle="-", marker='o', color='g', label='预测值')
        plt.legend()
    plt.show()
    def train_HistGradientBoostingRegressor(train_x, train_y, test_x, test_y, scaler=None):
        classifier = HistGradientBoostingRegressor()
        print('now training HistGradientBoostingRegressor...')
        classifier.fit(train_x, train_y)
        test_pred = classifier.predict(test_x)
        mse = mean_squared_error(test_y, test_pred)
        mae = mean_absolute_error(test_y, test_pred)
        rmse = mse ** 0.5
        r2 = r2_score(test_y, test_pred)
        print('MSE:', mse, 'MAE:', mae, 'RMSE:', rmse, 'R2:', r2)
        plt.figure()
        x = np.arange(1, len(test_y) + 1)
        plt.plot(x, test_y, linestyle=":", marker='o', color='b', label='真实值')
        plt.plot(x, test_pred, linestyle="-", marker='o', color='g', label='预测值')
        plt.legend()
    plt.show()

    问题 3 代码）
    if __name__ == "__main__":
        dill.load_session('pro1_1.pkl')
        # 变量标准化
        scaler_train = StandardScaler()
        scaler_train.fit(Mol_train_final_array)
        all_data = scaler_train.transform(Mol_train_final_array)
        all_label = ADMET_train_final_array
        # 采用相关性投票的方法选取变量
        [Caco_feature_array_weight, Caco_score_array] = seleimportpambyall(all_data,
    all_label[:, 0])
        [CYP3_feature_array_weight, CYP3_score_array] = seleimportpambyall(all_data,
    all_label[:, 1])
        [hERG_feature_array_weight, hERG_score_array] = seleimportpambyall(all_data,
    all_label[:, 2])
        [HOB_feature_array_weight, HOB_score_array] = seleimportpambyall(all_data,

all_label[:, 3])
        [MN_feature_array_weight, MN_score_array] = seleimportpambyall(all_data,
```

```
all_label[:, 4])
    if __name__ == "__main__":
        rcParams['axes.unicode_minus'] = False
        warnings.filterwarnings('ignore')
        mpl.rcParams['font.sans-serif'] = ['SimHei']
        plt.rcParams['font.sans-serif'] = ['SimHei']
        plt.rcParams['axes.unicode_minus'] = False
        dill.load_session('pro1_1.pkl')
        scaler_train = StandardScaler()
        scaler_train.fit(Mol_train_final_array)
        all_data = scaler_train.transform(Mol_train_final_array)
        all_label = ADMET_train_final_array
        all_test_xls = scaler_train.transform(Mol_test_final_array)
        train_data_all = []
        labels = []
        test_data_xls_all = []
        for i in range(5):
            train_data_all.append(all_data[:, weight_list[i]])
            test_data_xls_all.append(all_test_xls[:, weight_list[i]])
            labels.append(all_label[:, i])
        train_data = [[] for i in range(5)]
        test_data = [[] for i in range(5)]
        train_labels = [[] for i in range(5)]
        test_labels = [[] for i in range(5)]
        for i in range(5):
            train_data[i], test_data[i], train_labels[i], test_labels[i] =
train_test_split(train_data_all[i], labels[i],
    test_size=0.2,
    random_state=1)
        # 训练测试
        for i in range(5):
            for j in range(6):  # 第几种方法
                print([i, j])
                train_ML(train_data[i], train_labels[i], test_data[i], test_labels[i], scaler_train,
i, j)
```

4）问题四代码：

```
    if __name__ == "__main__":
        rcParams['axes.unicode_minus'] = False
        warnings.filterwarnings('ignore')
        mpl.rcParams['font.sans-serif'] = ['SimHei']
        plt.rcParams['font.sans-serif'] = ['SimHei']
        plt.rcParams['axes.unicode_minus'] = False
        dill.load_session('pro1_1.pkl')
```

```
        scaler_train = StandardScaler()
        scaler_train.fit(Mol_train_final_array)
        all_data = scaler_train.transform(Mol_train_final_array)
        all_label = ADMET_train_final_array
        all_label_ERa = ERa_train_final_array[:, 1]
        scaler_train_ERA = MinMaxScaler()
        scaler_train_ERA.fit(all_label_ERa.reshape(-1, 1))
        [weight_list_vote, wight_sore_list] = selewightbyvote(weight_list_pro2,
weight_list_pro3)
        train_data_all = []
        labels = []
        for i in range(5):
            train_data_all.append(all_data[:, weight_list_pro3[i]])
            # test_data_xls_all.append(all_test_xls[:, weight_list[i]])
            labels.append(all_label[:, i])
        train_data_all_ERa = all_data[:, weight_list_pro2]
        labels_ERa = all_label_ERa
        train_data = [[] for i in range(5)]
        test_data = [[] for i in range(5)]
        train_labels = [[] for i in range(5)]
        test_labels = [[] for i in range(5)]
        # 训练集和测试集
        for i in range(5):
            train_data[i], test_data[i], train_labels[i], test_labels[i] =
train_test_split(train_data_all[i], labels[i],
    test_size=0.2,
    random_state=1)
        train_data_era, test_data_era, train_labels_era, test_labels_era =
train_test_split(train_data_all_ERa, labels_ERa,
    test_size=0.2,
    random_state=1)
        classifier_save = [[] for i in range(5)]
        for i in range(5):  # 找里面好的方法
            j = [6, 6, 2, 4, 2][i]
            classifier_save[i] = classifier_ML(train_data[i], train_labels[i], i, j)
        Reg_save = Regressor_ML(train_data_era, train_labels_era)
        bestEra = 0
        bestind = 0
        bestlist = [1, 0, 0, 1, 0]
        for i in range(len(all_label_ERa)):
            count = 0
            for j in range(5):
                if all_label[i][j] == bestlist[j]:
                    count += 1
```

```
if count > 3 and all_label_ERa[i] > bestEra:
    bestind = i
    bestEra = all_label_ERa[i]
```