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	上海海事大学
参赛队号	21102540518
队员姓名	1.吴嘉俊
	2.王蕾
	3.王宏宇

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 抗乳腺癌候选药物的优化建模

摘 要：

乳腺癌是目前世界上最为常见且致死率较高的癌症之一，而研究表明抑制雌激素受体 α 亚型（Estrogen receptors alpha，*ERα*）活性对治疗乳腺癌具有重大意义。为研发抑制 *ERα* 活性的有效新药物，现结合 QSAR 模型对新化合物分子的生物活性进行预测并优化，同时要保证新化合物的药代动力学性质和安全性。本文充分运用各种特征工程方法、数据挖掘技术、机器学习方法以及启发式算法，研究了如何对化合物的分子描述符优化取值，以获得药物更好的生物活性（此处指抑制 *ERα* 活性）以及药代动力学性质和安全性。

针对问题一：为确保特征选择的合理性，首先要对“ *ERα* _activity.xlsx”和“ *ERα* _activity.xlsx”文件中的数据进行预处理，剔除分子操作符中的无效变量，即数值完全一致的分子描述符，来减少特征选取时的干扰；其次检查数据中的异常值，由于所有自变量均为用于描述化合物性质和结构的客观参数，因此不对异常值进行特别处理；随后对低类别数据（即 0，1，2）进行分析，发现其并不适合处理为分类变量进行编码，且使用哑变量进行处理后的数据也并不影响特征选择的结果；最后选取 Z-zeros 标准化的方式对数据进行标准化处理。考虑使用 8 种特征选择方式对预处理后的数据进行特征选择，综合选取出现在 8 种方法排名前 50 出现频率最高的 20 个分子描述符作为最终选取的结果。

针对问题二：为建立化合物对 *ERα* 生物活性的定量预测模型，需要首先对问题一中的 20 个具有显著影响的分子描述符进行筛选，使用距离相关系数进行特征之间的相关性分析，选择剔除相关性较强的变量，筛选后剩余的 15 个分子描述符作为回归预测模型的输入特征。选取 80%的样本作为训练集，20%的样本作为测试集，使用 9 种回归预测方法进行模型训练和预测，并进行十折交叉验证分析，选择其中较优的三种回归方法：随机森林、GBDT、XGBoost，再分别对待测数据进行回归预测，进一步对预测效果进行分析比较，发现随机森林回归预测模型的决定系数为 0.76，均方误差为 0.45，证明其具有较好的预测精度，因此最终选用随机森林回归方法对“ *ERα* _activity.xlsx”文件“test”表中的 IC_{50}_nM 与 pIC_{50} 值进行预测得到结果。

针对问题三，为构建化合物的 Caco-2、CYP3A4、hERG、HOB、MN 分类预测模型，首先重新构建 5 个独立数据集，而题目给出的数据样本变量间可能存在高度非线性关系，因此还需要先对数据样本进行特征选择。考虑 9 种常用的预测模型进行分析对比，选取 80%的样本作为训练集，剩余 20%作为测试集对模型进行训练，分析预测结果的混淆矩阵、ROC 曲线线下面积、正确率等统计指标，最终选择 MLP 作为 CYP3A4 的最优预测模型，XGBoost

作为 Caco-2、hERG、HOB、MN 的最优预测模型，并利用训练好的 5 个模型分别对“test”表中的 ADMET 性质进行预测。

针对问题四：首先为寻找能够使化合物生物活性更好的主要分子描述符，使用了问题一中的 8 种特征选择方法，分别针对 pIC_{50} 以及 5 个 ADMET 性质进行特征选择，分别得到 8 个方法对 pIC_{50} 以及 5 个 ADMET 性质的分子描述符重要性排序，随后统计 6 个指定指标（pIC_{50} 以及 5 个 ADMET 性质），在 8 种方法重要性排名前 50 个变量出现的总频率，选取排名靠前的 20 个变量，再综合选取 6 个指定指标中共同出现频率最高的 20 个分子描述符作为特征选择的结果。然后使用这 20 个对所有指定指标都具有显著影响的分子描述符，对指标分别构建 6 个随机森林预测模型。再以最优化 pIC_{50} 的值作为优化目标，ADMET 性质、变量合理范围等作为约束条件建立数学优化模型，使用遗传算法进行优化求解，并使用之前所构建的 6 个预测模型对每一代的适应度和 ADMET 约束条件进行预测，最终得到最优的 pIC_{50} 值为 8.341，虽然其 ADMET 性质中 hERG 和 HOB 未达到较优效果，但其余 3 项性质均达到较优性质。

关键词：QSAR 模型；特征工程；随机森林；机器学习；遗传算法

目录

一、问题重述

1.1 研究背景

乳腺癌已经成为女性最常见的恶行肿瘤疾病,全世界的医学家都在为克服乳腺癌进行着不懈的斗争，而乳腺作为性器官之一，性激素与乳腺癌的关系非常密切，乳腺癌的内分泌治疗在临床上的成效已经得到了公认[1]。目前临床针对乳腺癌的一项重要治疗就是抗激素治疗，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。比如，临床治疗乳腺癌的经典药物他莫昔芬和雷诺昔芬就是 *ERα* 拮抗剂。但现有临床治疗药物并不能完全阻止乳腺癌的复发与转移,甚至常引起患者心血管损伤、免疫力低下、药物敏感性降低、绝经后诸症风险增加等诸多不良反应[2]。筛选安全高效的抗乳腺癌化合物活性成分,用以研发新型 *ERα* 拮抗剂是临床治疗乳腺癌的重要研究方向。文献[2]从生物活性多糖的生物特性、抗肿瘤活性构效关系出发，研究了多糖对乳腺癌的作用机制。文献[3]对 OBHS-HDACi 缀合物、OBHSA -HDACi 级合物和 FcOBHS-HDACi 级合物依次进行了合成和抗乳腺癌活性研究,基于此设计了雌激素受体和组蛋白乙酰结合的新型双靶点抗乳腺癌药物[3]。

能够拮抗 *ERα* 活性的化合物具有复杂多样性，但传统的化合物研究对象多为少样本化合物或单一化合物，很少有研究通过多样本 *ERα* 的生物活性数据综合多种化合物进行分析。因此，本文将立足于探索研究多种化合物对 *ERα* 活性的影响，构建化合物的定量结构-活性关系（Quantitative Structure-Activity Relationship, QSAR）模型及化合物的分类预测模型，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，以对乳腺癌治疗、药物研发等产生理论支撑。

随着信息技术的发展，数据挖掘技术已广泛应用于各个领域，其中也包含医疗领域，数据挖掘技术是从数据中发现所蕴含的规律，是通过建立数据驱动的数学模型来优化操作条件的重要方法[4]。因此，在寻找治疗乳腺癌候选药物过程中，亟需通过数据挖掘技术处理大量数据并设计精准的预测模型，以寻求与化合物的生物活性具有强相关性的分子结构描述符，在节约时间和成本的前提下，寻找能够拮抗 *ERα* 活性的化合物可能是治疗乳腺癌的候选药物，并保证其具备良好的抗乳腺癌活性、药代动力学性质和安全性。

1.2 问题提出

基于上述研究背景，以题目给定的 1974 个化合物的 729 个分子描述符作为数据分析对象，通过数据挖掘技术来建立化合物生物活性的定量预测模型，并构建 5 种分类预测模型来优化化合物的操作变量条件，在预测过程中化合物同时也要满足候选药物的性能要求。(即具备良好生物活性的基础上也具备良好的药代动力学性质和安全性)

下面将对具体问题进行描述：

问题一:根据文件“Molecular_Descriptor.xlsx”和“ *ERα* _activity.xlsx”提供的数据，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）。

问题二:结合问题一，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并叙述建模过程。然后使用构建的预测模型，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，并将结果分别填入“ *ERα* _activity.xlsx”的 test 表中的 IC_{50}_nM 列及对应的 pIC_{50} 列。

问题三:利用“Molecular_Descriptor.xlsx”文件中提供的 729 个分子描述符，针对“ADMET.xlsx”中的 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。然后使用所构建的 5 个分类预测模型，对“ADMET.xlsx”中“test”表的 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：寻找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 $ER\alpha$ 具有更好的生物活性，同时具有更好的 ADMET 性质（给定的 5 个 ADMET 性质中，至少 3 个性质较好）。

二、模型假设

假设 1：所有样本的化合物生物活性值及分子描述符的数据客观正确；
假设 2：所有分子描述符均是可以独立操作改变的控制变量；
假设 3：在优化时认为所提出的预测模型结果近似准确

三、符号说明

序号	符号	含义
1	v_{ij}	RF 算法分裂特征的切分值
2	n_{left}、n_{right}	RF 算法分裂后左右子节点
3	MSE	平方平均误差
4	MAE	绝对平均误差
5	α	距离相关系数的阈值
6	φ	SVM 算法权重向量
7	b	SVM 算法偏置向量
8	θ	SVM 算法松弛因子
9	C	SVM 算法惩罚因子
10	μ	拉格朗日乘子
11	ρ	SVM 算法置信范围
12	λ	SVM 算法距离间隔
13	k	KNN 算法最邻近取值
14	v_l	MLP 算法上层隐含层输出
15	L	MLP 算法下层隐含层数量
16	f	MLP 算法非线性激活函数
17	t	MLP 算法真实值
18	β	MLP 算法激活函数

四、问题一：特征选择——寻找主要变量

4.1 问题一分析

本题要求针对 1974 个化合物的 729 个分子描述符，选出对生物活性具有显著影响的前 20 个变量。由于药物研发过程中化合物的复杂性、多样性等因素，分子描述符与化合物活性呈现出一种非线性关系，本问通过 8 种特征选择方法，对变量的重要性进行排序，最终综合所有方法选取 20 个变量作为影响化合物活性的主要变量。

该问题中，原始数据给出了 1974 个化合物和 729 个分子描述符以及每个化合物的生物活性值IC_{50}和pIC_{50}，而两个生物活性指标可以由下列数学公式相互转化：

$$IC_{50} = \frac{1000000000}{10^{pIC_{50}}}$$

通过对数据进行必要的预处理过程之后，为得到更具代表性的变量，本问题采用了过滤法、嵌入法、树模型等 8 种特征选择方法综合对变量进行重要性排序，然后将通过 8 种方法得到的变量重要性排序，分别取前 50 个变量进行交集查找，统计出现总频率次数前 20 的变量作为最终变量选择结果，思路流程如图 4-1。

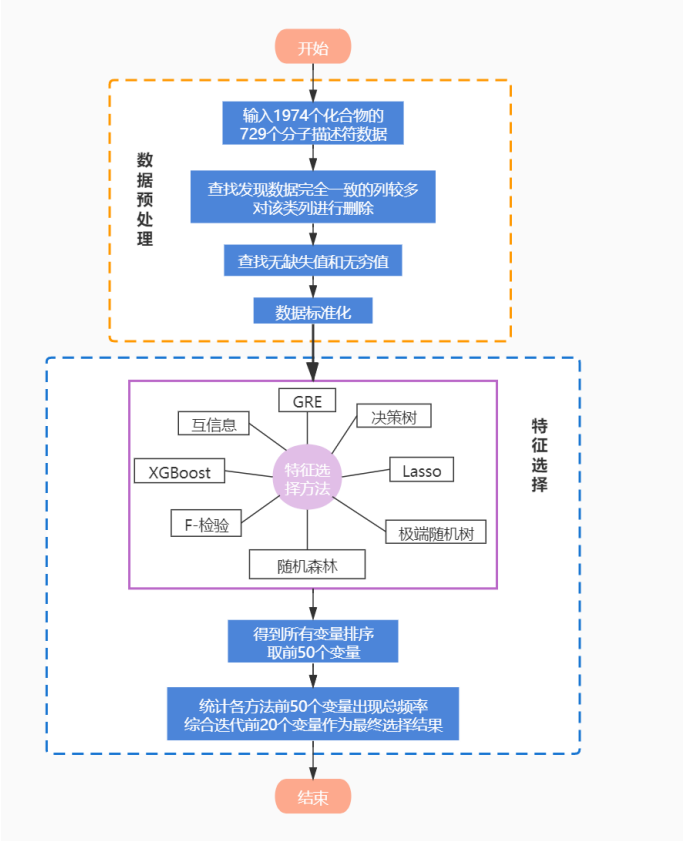


图 4-1 问题一思路流程图

4.2 数据预处理

4.2.1 剔除无效变量

经数据初步检查发现，原始数据中存在 225 个数值完全重复的分子描述符，该类变量在建模过程中无意义。直接删除该类变量，减少特征选取的干扰。被删除的变量如表 4-1

所示。

表 4-1 数值一致的分子描述符

	分子描述符
1	nB
2	nBondsQ
3	nHsNH3p
…	…
225	nF5Ring

4.2.2 异常值处理

首先对数据样本中 pIC_{50} 绘制了频数直方图如下，发现数据分布集中在[1,13], 且近似呈现正态分布，因此认为其数据合理，不做特别处理。

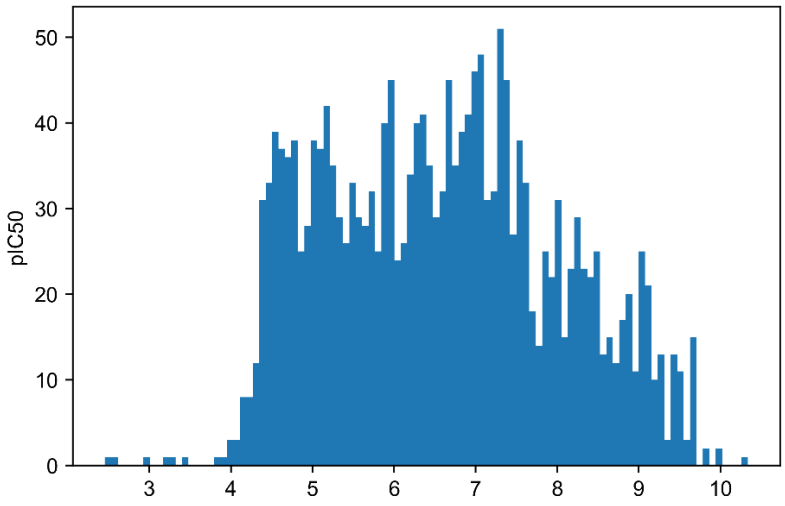


图 4-2 因变量 pIC_{50} 频数直方图

且由于所有自变量为一系列用于描述化合物结构和性质特征的客观参数，因此本文不对自变量进行异常值处理，认为所有自变量均处于合理取值范围内。

4.2.3 低类别数据

原始数据均为连续性数据，但通过观察发现部分变量尽管为连续变量，但实际总类别不超过 5 类，经查看“分子描述符. xlsx”文件后发现该类变量主要含义为“次数”、“个数”等，这类数据分布形式理论上服从泊松分布，不适合处理为分类变量进行编码。因此本文不对这类变量进行实际处理。

4.2.4 数据标准化

所有原始数据的分布箱线图如图 4-3 所示（由于多变量，所有箱线图均经过调整缩放），由于原始数据中每个分子描述符参数的含义各不相同且取值范围和量纲都不尽相同，从图中可以发现，原始数据取值范围大小各不相同且难以同时衡量。

因此，为了消除不同量纲对数据分析的影响，有必要对数据进行标准化处理。标准化的实质是对原始数据作出一种线性变换，在做数据分析时可以提高数据的表现。

标准化方法有两种，一种是 min-max 标准化，其转换函数为：

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

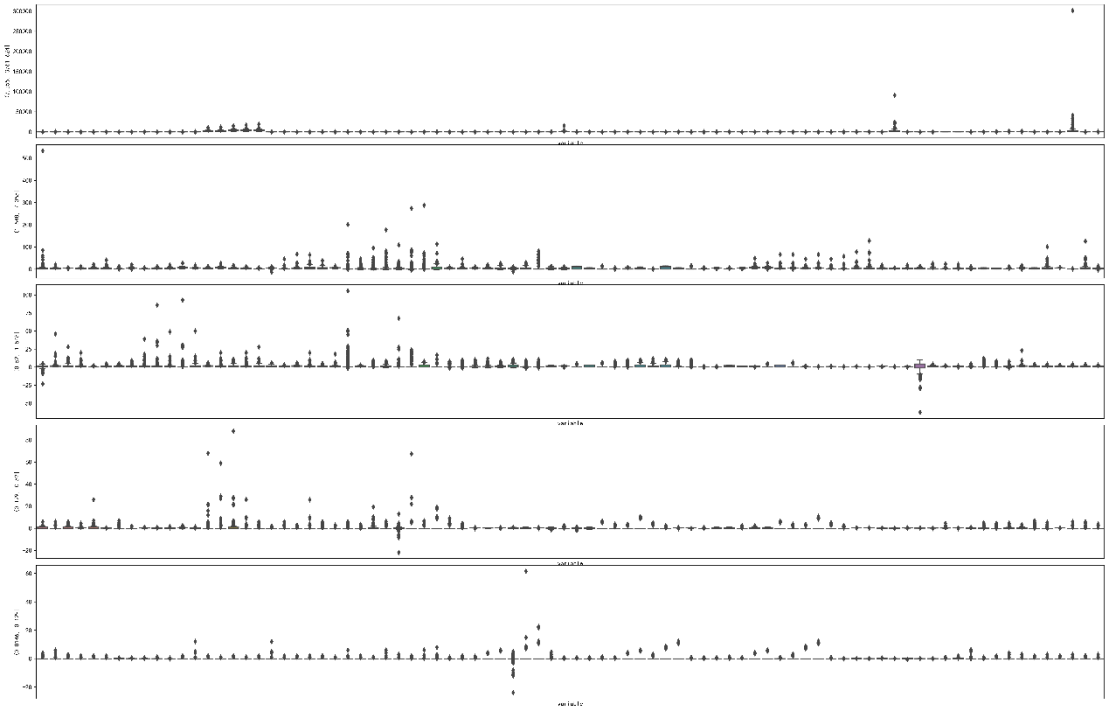


图 4-3 原始数据的分布箱线图

另一种标准化方法为 Z-score 标准化（0-1 标准化），Z-score 标准化会使每个变量中的数据平均值 μ 变为 0、标准差 σ 变为 1，该方法转换函数为：

$$x^* = \frac{x - \mu}{\sigma}$$

针对问题一中原始数据的特征，考虑采取 Z-score 标准化进行数据处理，标准化后的数据分布箱线图如下所示：

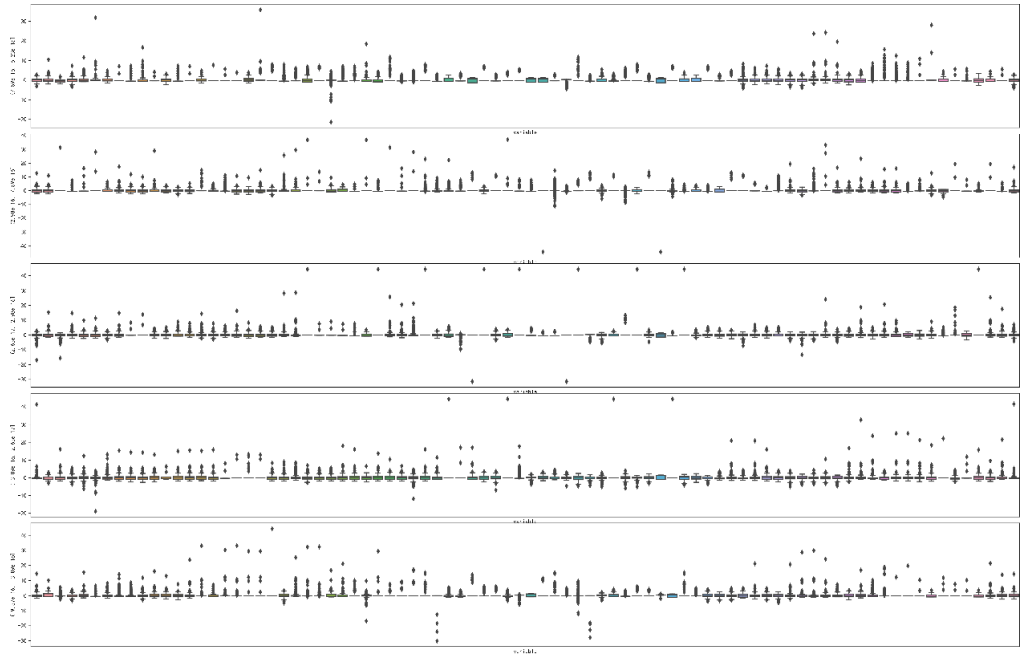


图 4-4 Z-score 标准化处理后分布箱线图

4.3 特征选择

考虑到分子描述符与生物活性之间的非线性关系，该问采用 F 检验过滤（FTest）、互信息(MutuaInfo)、灰色关联度(GRA)、决策树(DecisionTree)、随机森林(RandomForest)、极端随机树(ET)、L1 正则化(Lasso)等方法对变量的重要性进行排序，选取各方法重要性排序前 50 名的变量作为特征选择的结果。

（1）F 检验过滤法：描述两组数据之间的线性关系。

优点：既可以做回归也可以做分类；缺点：不能够找出变量间的非线性关系。

$$S^2 = \frac{\sum (X - X^-)^2}{n-1}$$

$$F = \frac{S^2}{S'^2}$$

（2）互信息法过滤法：

描述特征与标签之间的关系，返回[0,1]之间的互信息估计量，0 表示两个变量独立，1 则表示两个变量完全相关。

优点：可以找出任意线性或非线性关系，既可以做回归也可以做分类；缺点：不返回 p 值或 F 值类似的统计量。设 x,y 为两个随机变量，则有：

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

（3）灰色关联度 GRA：

分析各个自变量对因变量的影响程度，通过确定参考数据列和若干个比较数据列的几何形状相似程度来判断其联系是否紧密，它反映了曲线间的关联程度。

（4）决策树：

通过先验知识，计算信息增益，以此来划分分类点。决策树很容易过拟合，因此需要通过剪枝来降低过拟合。信息熵公式：

$$H(D) = -\sum_{k=1}^{k} p_k \log_2 p_k$$

其中，D 表示当前数据集合；k 表示当前数据集合中的第 k 类。

（5）随机森林：

随机森林的组成单元是决策树，通过集成学习思想将多棵决策树集成的算法随机森林集成了所有的分类投票结果，将投票次数最多的类别指定为最终的输出，这就是一种最简单的 Bagging 思想。特点：随机森林应用灵活，具有很好的准确率，能处理大规模高维数据特征样本，且能评估各个特征在分类问题上的重要性。

（6）极端随机树 ET：每棵决策树应用相同的全部训练样本，完全随机的得到分叉值，从而实现对决策树进行分叉的。

（7）Lasso 正则化：

缩小变量集的压缩估计方法。通过构造一个惩罚函数，可以将变量的系数进行压缩并使某些回归系数变为 0，进而达到变量选择的目的。

优点：可以产生稀疏特征因而一般有较好的表现。

通过使用上述 8 种特征选择方法，分别得到分子描述符的重要性指标排序，随后对重要性排名前 50 的变量进行可视化，进一步观察不同特征选择方法之间得到的变量重要性差异，绘制直方图如图 4-5 所示。图中横坐标表示变量重要性，纵坐标表示分子描述符。

从图中可以发现，F 检验过滤、互信息、GRA、Lasso 正则化的系数值分布较为均衡，走势平缓。而其他选择方法，如决策树、XGBoost、随机森林、极端决策树等方法，除了排名靠前的几个变量重要性水平比较明显，其他的变量重要性程度几乎为 0。其中，MDEC-23 在决策树、随机森林、极端决策树中均排在靠前位置，说明该变量对化合物活性的影响比较大。

进一步对各方法中重要性排前 50 的变量作进行筛选：对所有变量排名在前 50 的频数进行排序，综合得到问题一要求的前 20 个对生物活性由显著影响的分子描述符，处理结果如表 4-2 所示。

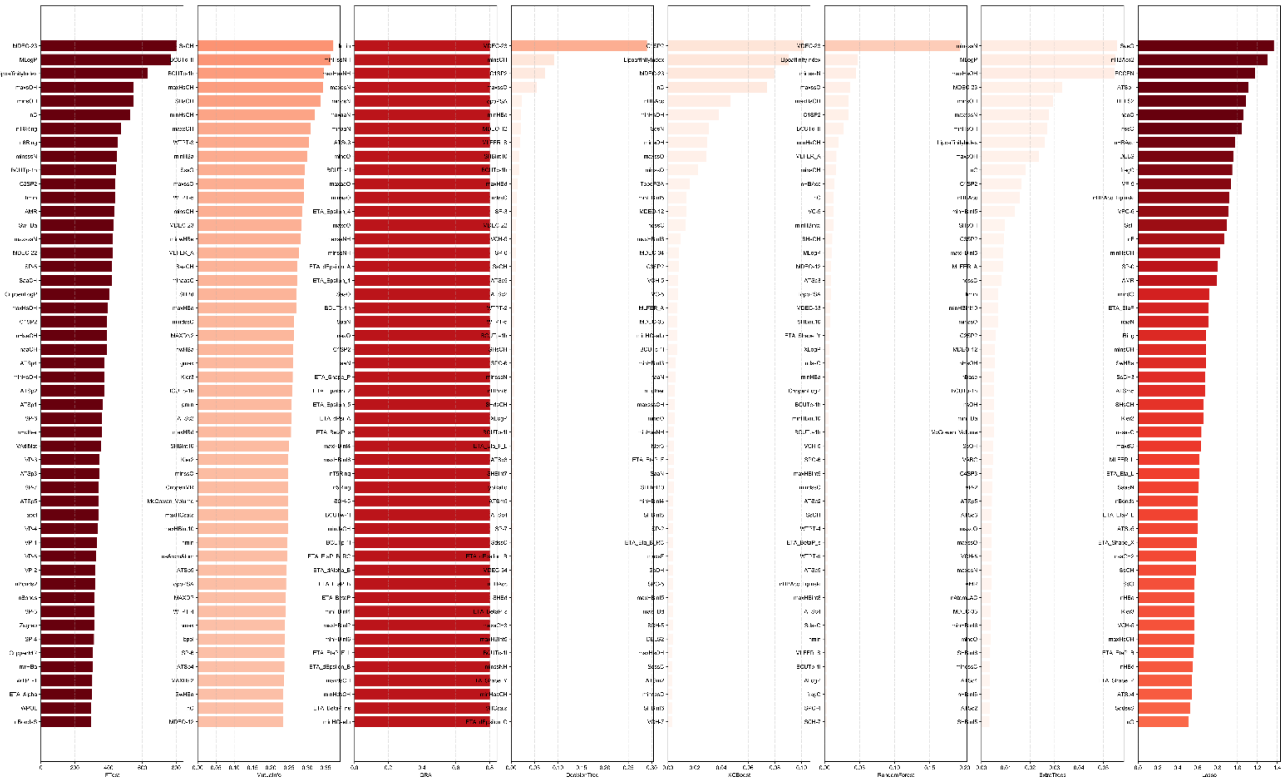


图 4-5 变量重要性排序直方图

表 4-2 20 个对生物活性由显著影响的分子描述符

Rank	Descriptor	Rank	Descriptor
1	minsOH	11	SHsOH
2	SsOH	12	VCH-5
3	maxHsOH	13	BCUTp-1h
4	BCUTc-1l	14	mindO
5	MDEC-23	15	hmin
6	C1SP2	16	maxssO
7	nHBAcc	17	minHBa
8	minHsOH	18	mindssC
9	ATSp5	19	nC
10	ATSc2	20	TopoPSA

五、问题二：化合物生物活性的定量预测模型

5.1 问题二分析

问题二要求建立化合物对 *ERα* 生物活性的定量预测模型。结合问题一中分子描述符选择结果，再通过机器学习和数据挖掘技术建立 *ERα* 生物活性的预测模型，并进行模型预测效果的分析与验证。在该问中将考虑使用随机森林（Random Forest）、梯度提升决策树（GBDT）、极端梯度提升（XGBoost）三种预测模型算法分别构建预测模型，并加以验证比较，选择最优的预测模型对文件“ *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。针对问题二的要求，需要结合问题一的求解结果选出不超过 20 个变量，构建 *ERα* 生物活性的定量预测模型，并对预测模型进行验证分析证明其有效性与合理性，随后针对文件“ *ERα* _activity.xlsx”中 test 表的 50 个化合物进行 IC_{50} 值和对应的 pIC_{50} 值的预测。

通过对样本数据及分子描述符变量的观察与分析，考虑到特征两两之间可能存在较强相关性，因此需要对问题一的选择结果进行相关性分析，以筛选出符合问题要求的特征变量用于该预测模型，而分子描述符变量之间存在高度的非线性关系，因此，本问需要构建高维非线性回归预测模型对化合物进行预测。该问中将考虑选择三种常见的高效非线性回归预测模型算法，即：随机森林（Random Forest）、梯度提升决策树（GBDT）、极端梯度提升（XGBoost），并将三种算法加以分析和比较，选择其中预测回归效果最优的预测模型对文件“ *ERα* _activity.xlsx”的 test 表中的 50 个化合物进行 IC_{50} 值和对应 pIC_{50} 值的预测。

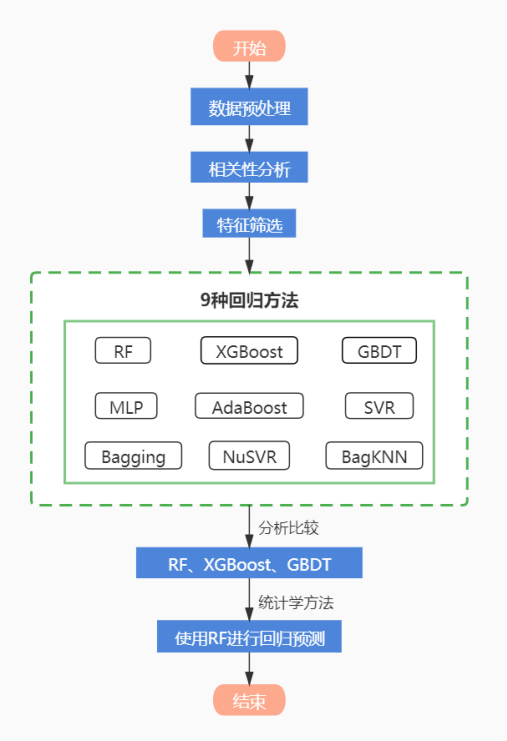


图 5-1 问题二思路流程图

问题二具体解题思路如图 5-1 所示，首先使用距离相关系数对第一问选出的变量进行相关性分析，筛选出合适的分子描述符。随后使用 9 种回归预测方法对“ *ERα* _activity.xlsx”中 50 个化合物的待测 IC_{50} 和 pIC_{50} 值进行预测，并分析比较预测效果，选择其中效果最优

的三种方法：随机森林、GBDT、XGBoost 进一步分析对比，最终选定随机森林作为本题回归预测的方法，对待测数据预测并求得结果。

5.2 构建化合物生物活性的定量预测模型

5.2.1 特征相关性分析

在问题一中选取了 20 个对生物活性最具显著影响的分子描述符，考虑到其中可能有变量两两之间存在强相关性，因此不适宜将 20 个分子描述符全部用到本题的预测模型中。检验变量相关性的常用方法为 Pearson 相关系数法，其仅适用于检验线性关系变量的相关性，而距离相关系数可解决非线性相关问题。针对该非线性问题，即考虑采用距离相关系数来判断变量之间的相关性，最终将相关性较大的变量选择筛除，剩余的变量作为输入特征应用到本题预测模型中进行预测分析。

距离相关系数即是将两个随机变量的距离协方差除以它们的距离标准差的乘积，即可得到这两个变量的距离相关系数。设 $\{(u_i,v_i)=0,i=1,2,...,n\}$ 为总体 (u,v) 中观察到的随机样本，则 u,v 之间的距离相关系数估计值为：

$$\partial corr(u,v)=\frac{\partial \operatorname{cov}(u,v)}{\sqrt{\partial \operatorname{cov}(u,u)\partial \operatorname{cov}(v,v)}}$$

其中 $\partial \operatorname{cov}^2(u,v)=\hat{S}_1+\hat{S}_2-2\hat{S}_3$，$\hat{S}_1$、$\hat{S}_2$ 和 $\hat{S}_3$ 分别为：

$$\hat{S}_1=\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left\|u_i-u_j\right\|_{d_u}\left\|v_i-v_j\right\|_{d_v}$$

$$\hat{S}_2=\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left\|u_i-u_j\right\|_{d_u}\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left\|v_i-v_j\right\|_{d_v}$$

$$\hat{S}_3=\frac{1}{n^3}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\left\|u_i-u_j\right\|_{d_u}\left\|v_i-v_j\right\|_{d_v}$$

当距离相关系数较小时，即两个变量间的相关性较小，当距离相关系数趋近于 1 时，则两个变量间存在强相关性。

距离相关系数的强弱度量考虑如表 5-1 所示：

表 5-1 相关系数

相关程度	相关系数
极强相关	0.8—1.0
强相关	0.6—0.8
中相关	0.4—0.6
弱相关	0.2—0.4
极弱相关	0.0—0.2

为了提高优化后预测模型的精准度，需设定距离相关系数的阈值，作为各输入特征之间距离相关系数的判定标准，本问将其设定为 $\alpha=0.8$。若分子描述符两两之间的距离相关系数大于 α，判定变量之间具有较强的相关性，则选择二者之间在问题一中排名更高的特征

进行保留，剔除另一个。计算后得到问题一中 20 个分子描述符两两之间的距离相关系数分布图如图 5-2 所示，其中颜色越深代表变量之间的相关性越大，需要对其进行选择筛选。

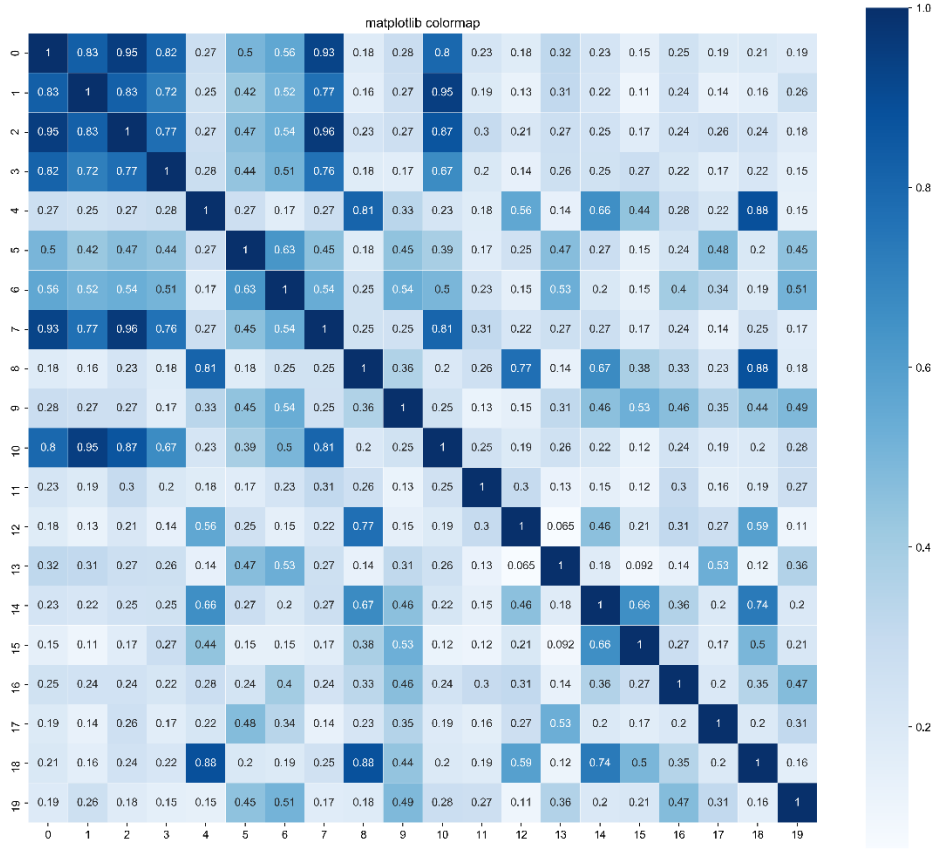


图 5-2 距离相关系数分布图

经过筛选后，保留 15 个分子描述符作为特征，如表 5-2 所示：

表 5-2 保留的分子描述符及特征

NO.	Descriptor	Description	Class
22	ATSc2	ATS autocorrelation descriptor, weighted by charges	2D
39	BCUTc-1l	nhigh lowest partial charge weighted BCUTS	2D
42	BCUTp-1h	nlow highest polarizability weighted BCUTS	2D
56	C1SP2	Doubly hound carbon bound to one other carbon	2D
70	VCH-5	Valence chain, order 5	2D
291	SsOH	Sum of atom-type E-State: -OH	2D
346	minHBa	Minimum E-States for (strong) Hydrogen Bond acceptors	2D
392	mindssC	Minimum atom-type E-State: =C<	2D
410	minsOH	Minimum atom-type E-State: -OH	2D
411	mindO	Minimum atom-type E-State: =O	2D
476	maxHsOH	Maximum atom-type H E-State: -OH	2D
531	maxssO	Maximum atom-type E-State: -O-	2D
585	hmin	Minimum H E-State	2D
639	nHBAcc	Number of hydrogen bond acceptors (using CDK HBondAcceptorCountDescriptor algorithm)	2D
716	TopoPSA	Topological polar surface area	2D

5.3 模型验证与分析

为选择合适的回归预测模型，现分别使用 9 个常用的回归预测方法（随机森林、XGBoost、AdaBoost、GBDT、多层感知机、支持向量回归、引导聚集算法、KNN、NuSVR）针对“*ERα*_activity.xlsx”文件和“Molecular_Descriptor.xlsx”文件的样本数据集，以 80%的样本作为训练集，20%的样本作为测试集，进行十折交叉验证，得分结果如下表所示：

表 5-3 交叉验证结果

Num	RF	XGBoost	AdaBoost	GBDT	MLP	SVR_rbf	Bagging	Bag_KNN	NuSVR
1	0.722	0.692	0.545	0.702	0.585	0.655	0.698	0.657	0.653
2	0.686	0.637	0.527	0.633	0.598	0.634	0.650	0.623	0.639
3	0.734	0.685	0.588	0.688	0.614	0.649	0.703	0.673	0.657
4	0.723	0.690	0.561	0.686	0.620	0.664	0.698	0.688	0.663
5	0.706	0.665	0.531	0.667	0.579	0.653	0.687	0.650	0.646
6	0.673	0.666	0.521	0.646	0.597	0.617	0.640	0.628	0.629
7	0.723	0.708	0.475	0.660	0.593	0.654	0.702	0.667	0.649
8	0.713	0.711	0.551	0.673	0.642	0.667	0.685	0.648	0.664
9	0.719	0.693	0.550	0.685	0.453	0.661	0.687	0.650	0.657
10	0.738	0.712	0.570	0.691	0.661	0.691	0.704	0.689	0.695

通过绘制箱线图进行可视化处理，如图 5-3 所示。

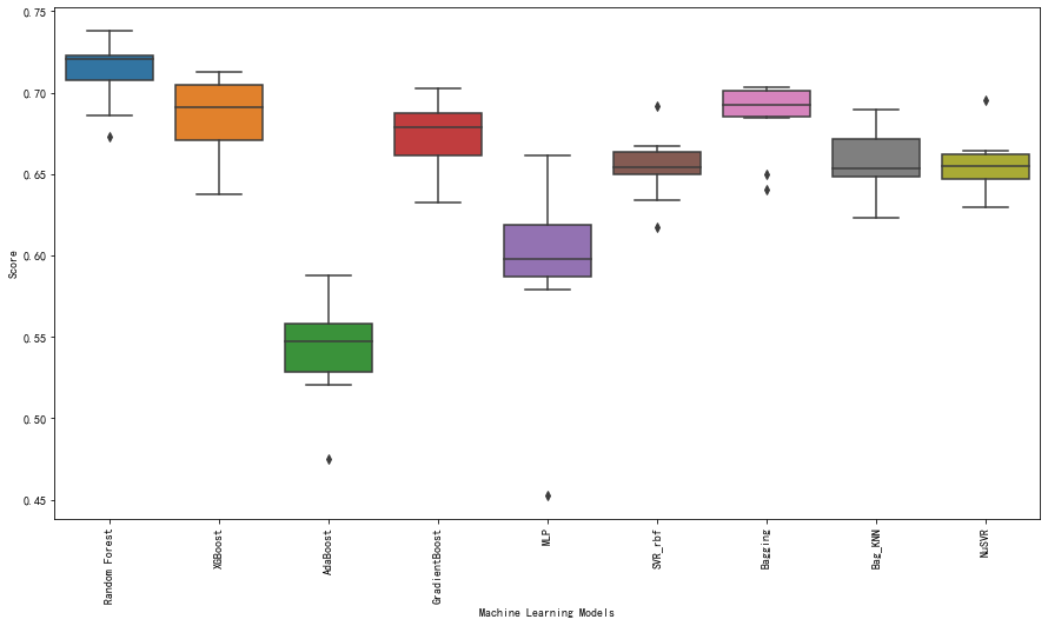


图 5-3　回归分析模型箱线图

可以直观的看出随机森林、GBDT、XGBoost 这三种回归预测方法均具有较好的回归预测效果。因此接下来主要对比随机森林、GBDT、XGBoost 这三种方法对数据的预测效果。

5.3.1 基于随机森林的定量预测模型

随机森林（Random Forest）是利用多棵决策树针对样本数据进行训练，并能够进行分类与预测的一种机器学习方法，属于集成学习中 Bagging 方法的一种。其通过集成学习的思想将 N 棵决策树集成，预测模型建立之后，当有新的测试数据进入随机森林，其中的每一棵决策树会分别对新的测试数据进行独立的判断，在分类问题中往往采用最大投票数的

方法，得票最多的类别标签将被当作结果输出，而在回归预测问题中则使用简单平均法，将 K 个决策树的回归结果进行算术平均处理，处理后的结果即作为最终输出结果[5]。

随机森林中的每棵树将按照如下的规则生成：

（1）若训练集的大小为 N，随机森林中每棵树都随机且有放回地每次抽取 1 个训练样本，共抽取 N 次，作为每棵决策树的训练集，这种方法也被称为 Bootstrap sample 方法。这使得每棵树的训练集样本是不同的，因此能够保证每棵决策树的独立性。

（2）若每个数据样本的特征维度为 D，则指定常数 d<<D，从 M 个特征中随机抽取 m 个特征子集，在每次决策树进行分裂步骤时，从中选取最优的特征作为分裂节点。

（3）随机森林中每棵决策树尽可能的生长，并且不做修剪。

对随机森林的训练即是对 K 个决策树的训练，需要考虑的关键点在于如何选择特征、分裂点以及怎样判断特征和分裂点的好坏。对于此，一般采用分裂后节点的不纯度来进行衡量判断，即各子节点不纯度的加权和$G(x_i, v_{ij})$，则有：

$$G(x_i, v_{ij}) = \frac{n_{left}}{N_s} H(X_{left}) + \frac{n_{right}}{N_s} H(X_{right})$$

其中x_i为某一个分裂特征，v_{ij}为分裂特征的切分值，n_{left}，n_{right}，N_s分别为分裂后左右子节点的训练样本个数以及当前节点的所有训练样本个数，X_{left}，X_{right}分别为分裂后左右子节点的训练样本集合，$H(X)$为衡量节点不纯度的函数，对于回归预测问题$H(X)$常使用平方平均误差（MSE）以及绝对平均误差（MAE）

$$\text{MSE：}\quad H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y - \bar{y}_m)^2$$

$$\text{MAE：}\quad H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} \left| y - \bar{y}_m \right|$$

该问中将考虑选择 MSE 作为不纯度函数，来进行决策树的分裂与训练，即针对每个分裂点有：

$$G(x, v) = \frac{1}{N_s} \left(\sum_{y_i \in X_{left}} (y_i - \bar{y}_{left})^2 + \sum_{y_i \in X_{right}} (y_i - \bar{y}_{right})^2 \right)$$

如图 5-4 所示是随机森林中单个决策树的训练过程，首先根据 Bootstrap sample 方法抽取 N 个训练样本，再在 D 个特征中随机抽取 d 个特征子集（常有$d = \log_2 D$），从中选取最优特征作为分裂节点进行分裂步骤，由此生成决策树中的单个决策树。

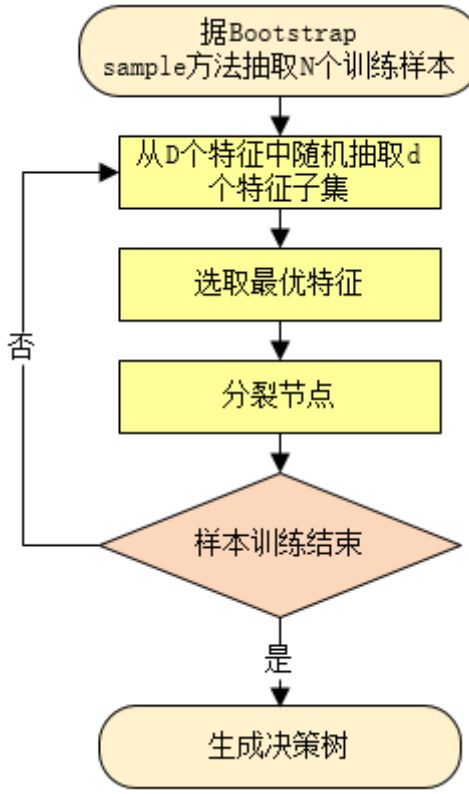


图 5-4 随机森林单个决策树训练过程

如图 5-5 所示是随机森林回归预测模型的预测过程，将每棵树的预测结果最后进行简单平均处理作为最终的输出预测结果。

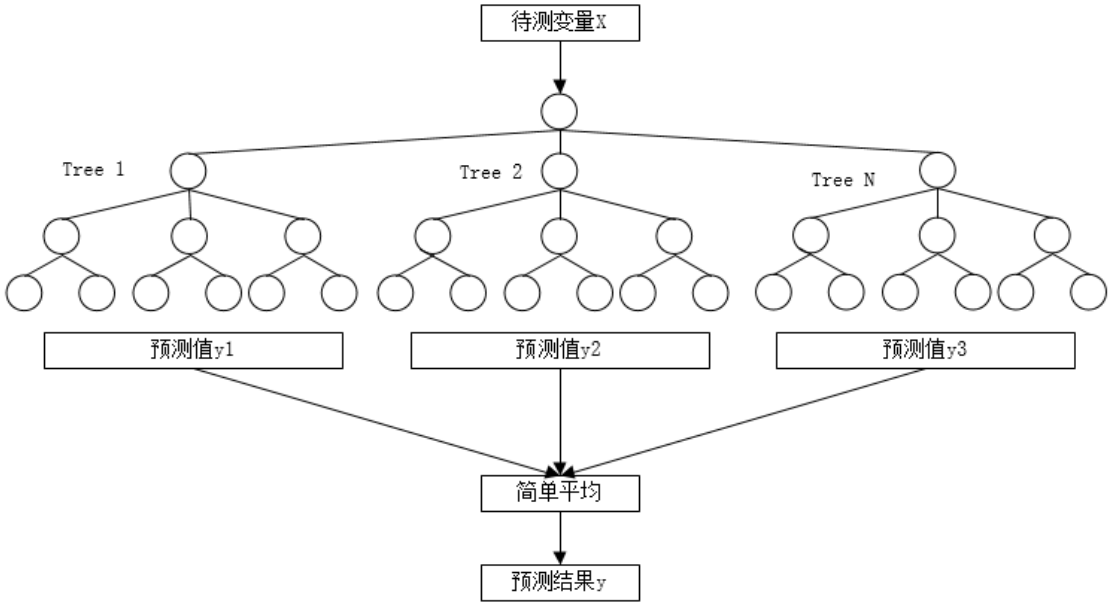


图 5-5 随机森林回归预测模型的预测过程

随机森林算法能够有效地在大数据集上运行，而且引入了随机性不容易导致过拟合的现象，同时其具有很好的抗噪能力，每棵树具有独立性可以很好的做成并行化方法，对回归预测问题的预测精度较高。

使用随机森林对训练集与测试集预测的效果（决定系数和均方误差）如下表所示，对

测试集预测的残差图绘制如图 5-6 所示，可以看到随机森林整体的预测效果均较为优良，且残差图中的异常值波动也较少，具有良好的预测效果。

表 5-4 随机森林预测效果

评价指标	训练集	测试集
决定系数	0.94	0.76
均方误差	0.13	0.45

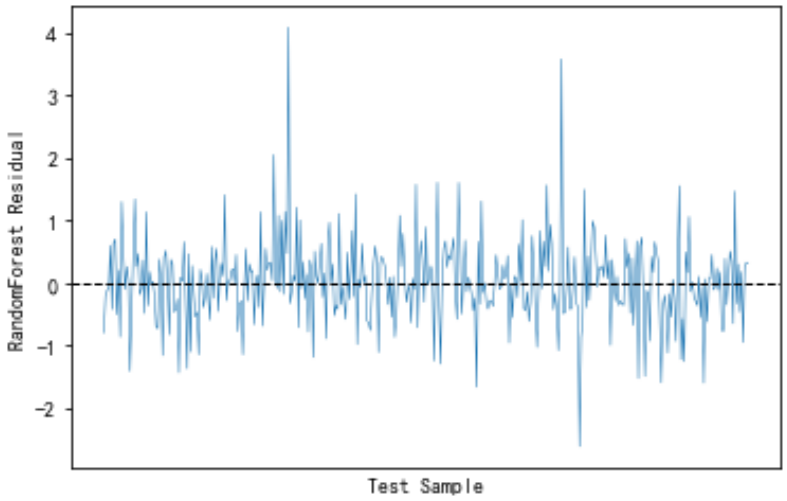


图 5-6 随机森林预测残差图

5.3.2 基于 GBDT 的定量预测模型

梯度提升决策树（GBDT）是传统经典机器学习算法中对于真实分布拟合最好的算法之一，是一种迭代的决策树算法，该算法与随机森林类似也是由多棵决策树所组成，但不同之处在于 GBDT 中的决策树为回归树，而不是随机森林中的分类树，回归树的每一个节点得到的是一个预测的量化值，节点分裂时将不再使用最大熵的条件，而是最小化平方误差。构建回归树的步骤如下：

在训练数据集的空间中，使用递归的方法将每个区域划分为两个子区域，并预测每个子区域的输出值由此来构建回归树。

（1）首先选择最优的分裂特征 j 与分裂点 s，求解：

$$\min_{j,s}\left[\min_{c_1}\sum_{x_1\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_2\in R_2(j,s)}(y_i-c_2)^2\right]$$

（2）应用选定的(j,s) 划分子空间并决定相应的输出值：

$$R_1(j,s)=x\,|\,x^j\le s, R_2(j,s)=x^j\ge s$$

$$c_m=\frac{1}{N_m}\sum y_i$$

（3） 继续重复（1）（2）直至满足终止条件

（4）将输入空间划分为 M 个子空间 $R_1,R_2,...,R_M$，生成回归树：

$$f(x)=\sum_{m=1}^{M}c_m I(x\in R_m)$$

而对于 GBDT 方法中另一个关键的思想即是提升树算法思想，即是在算法中迭代 N 棵回归树来进行回归决策，而第 n($n\in\{1,2.,..,N\}$)棵树将从之前所有 n-1 棵树的结果与残差中得到提升与学习，通过拟合的方式生成当前残差回归树，其中残差=真实值-预测值

GBDT 即是将回归树与提升树的思想相结合，其算法主要步骤如下：

（1）将训练数据集、损失函数等进行初始化，如下：

$$f_0(x)=\arg\min_{\gamma}\sum_{i=1}^{n}L(y_i,\gamma)$$

（2）进行 M 次迭代生成 M 棵决策树

i. 计算残差近似值

$$r_{im}=-\left[\frac{\partial L(y_i,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}}$$

ii. 根据残差近似值拟合为决策树

iii. 计算叶节点域的值

$$\gamma_{jm}=\underbrace{(argmin)}_{\gamma}\sum_{x_i\in R_{jm}}L(y_i,f_{m-1}(x_i)+\gamma)$$

iv. 子决策树更新

$$f_m(x)=f_{m-1}(x)+\sum_{j=1}^{J_m}\gamma_{jm}I(x\in R_{jm})$$

（3）输出最终值

$$\hat{f}(x)=f_M(x)=f_0(x)+\sum_{m=1}^{M}\sum_{j=1}^{J}\gamma_{jm}I(x\in R_{jm})$$

GBDT 能够有效地处理非线性回归问题，且在较少的调参操作下，仍能有较高的预测精度。

使用 GBDT 对训练集与测试集预测的效果如下表所示，对测试集预测的残差图绘制如图 5-7 所示，可以看到 GBDT 整体的预测效果均较不算优秀，且残差图中的异常值波动也较大，拥有较多异常点。

表 5-5 GBDT 预测效果

评价指标	训练集	测试集
决定系数	0.91	0.74
均方误差	0.19	0.48

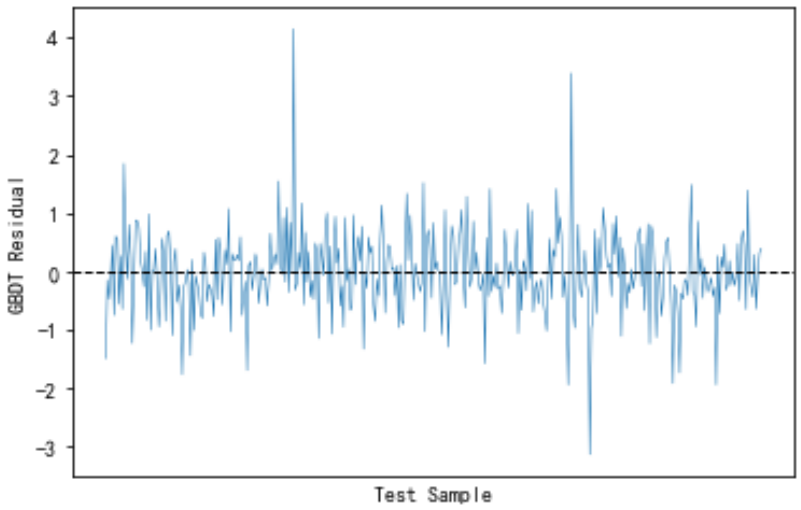


图 5-7 GBDT 预测残差图

5.3.3 基于 XGBoost 的定量预测模型

极端梯度提升（XGBoost）是一种在 GBDT 算法的基础上加以改进的机器学习方法，常被用于解决大规模数据挖掘问题。其运用集成学习的思想，应用加法模型和前向分布模型将 N 个回归树集成，并对各回归树评价选择，以此来实现回归预测[6]。

相对于 GBDT 算法，XGBoost 在优化过程中对代价函数进行了二阶泰勒展开并加入了正则化项以防止过拟合的现象，因此在使用训练集数据对其进行训练时其目标函数由梯度提升损失以及正则化项共同组成。其损失函数定义如下：

$$L(\Phi)=\sum_{i=1}^{n} l(y_i', y_i)+\sum_{k}\Omega(f_k)$$

XGBoost 算法的回归预测过程如下：

（1）现有 N 个训练集数据样本，对每个训练集数据样本构建回归树，则有 N 棵回归树，如下所示：

（2）$\hat{y}_i$ 表示第 i 个训练样本的预测值，$\sum_{i=1}^{n} f_k(x_i)$ 表示第 i 个变量的第 k 个回归树对应的回归方程：

$$\hat{y}_i=\sum_{i=1}^{n} f_k(x_i),\ f_k\in F$$

（3）建立目标函数即由梯度提升损失函数以及正则化项组成。

（4）将损失函数经泰勒展开得到近似真实值，如下所示：

$$L(y_i,\hat{y}_i^{(t-1)}+f_t(x_i))=L(y_i,\hat{y}_i^{(t-1)})+g_i f_t(x_i)+\frac{1}{2}h_i f_t^{2}(x_i)$$

（5）最后得到目标函数：

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

XGBoost 其具有的正则化项可以有效防止过拟合现象，且应用了泰勒展开式使得损失函数更加精确，能够高效地对非线性回归问题进行预测分析。

使用 XGBoost 对训练集与测试集预测的效果如下表所示，对测试集预测的残差图绘制如图所示，可以看到虽然 XGBoost 对于训练集样本的预测效果最佳，但对于待测数据集的预测残差波动较大，存在较多异常值。

表 5-6 回归预测方法预测效果分析

评价指标	训练集	测试集
决定系数	0.98	0.74
均方误差	0.03	0.48

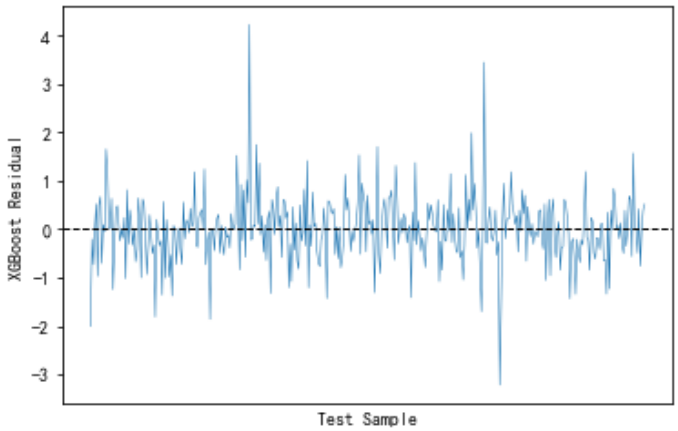


图 5-8 XGBoost 预测残差图

综合以上考虑，随机森林算法对于该问中的训练集及待测数据集均具有较为优良的预测效果，且预测残差波动异常值较少，因此最终选择随机森林回归预测模型对该问的待测数据集进行预测求解。

5.4 模型求解

随机森林预测模型参数设置如表 5-7 所示：

表 5-7 预测模型参数设置

名称	符号	取值
叶子节点最小样本数	min_samples_leaf	2
叶子节点最小样本权重	min_weight_fraction_leaf	0
子树继续划分阈值	min_samples_split	4
随机森林中分类器个数	n_estimators	300
随机数设置	Random_state	1234
bootstrap	bootstrap	True

使用随机森林回归方法对“*ERα*_activity.xlsx”文件中“test”表的 50 个待测 IC50_nM 和 pIC_{50} 值进行回归预测，预测结果如表 5-7 所示：

表 5-8 模型预测结果

Num	IC50_nM	pIC_{50}	Num	IC50_nM	pIC_{50}
0	120.222	6.920	25	13025.190	4.885
1	197.082	6.705	26	3694.247	5.432
2	191.324	6.718	27	189.860	6.722
3	248.152	6.605	28	206.467	6.685
4	18.977	7.722	29	7666.521	5.115
5	454.494	6.342	30	10827.750	4.965
6	234.560	6.630	31	2430.007	5.614
7	413.272	6.384	32	2483.217	5.605
8	124.425	6.905	33	2425.235	5.615
9	483.735	6.315	34	11805.390	4.928
10	306.568	6.513	35	10335.280	4.986
11	427.229	6.369	36	10281.640	4.988
12	412.000	6.385	37	1134.019	5.945
13	539.400	6.268	38	1337.461	5.874
14	476.580	6.322	39	1189.993	5.924
15	521.970	6.282	40	441.030	6.356
16	363.394	6.440	41	396.485	6.402
17	194.336	6.711	42	1282.666	5.892
18	396.409	6.402	43	1430.599	5.844
19	127.955	6.893	44	441.030	6.356
20	202.831	6.693	45	178.323	6.749
21	174.140	6.759	46	104.569	6.981
22	15.747	7.803	47	121.736	6.915
23	19.880	7.702	48	124.173	6.906
24	439.989	6.357	49	127.725	6.894

六、问题三：分类预测模型

6.1 问题三分析

问题三要求根据题目给出的数据分别构建化合物的 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型。从数据类型来看，变量间可能存在高度非线性关系。为得到 5 个化合物 ADMET 性质的分类预测模型，将题目所给的数据重新整理为 5 个独立数据集。每个数据集取 80%样本作为训练集，剩余 20%样本作为测试集用于模型评价与验证。通过在训练集上建立分类模型，然后在测试集上预测模型。训练集共 1572 个观测样本，测试集共 395 个观测样本。

通过比较 AdaBoost、KNN、决策树（DT）、逻辑回归（LR）、GBDT、XGBoost、支持向量机（SVM）、随机森林（RF）、MLP 等分类预测方法对独立数据集的分类预测效果，根据每个分类模型在数据集中的正确率、混淆矩阵及 ROC 曲线下面积，选出每个数据集的最佳分类预测模型。问题三的思路流程如图所示。

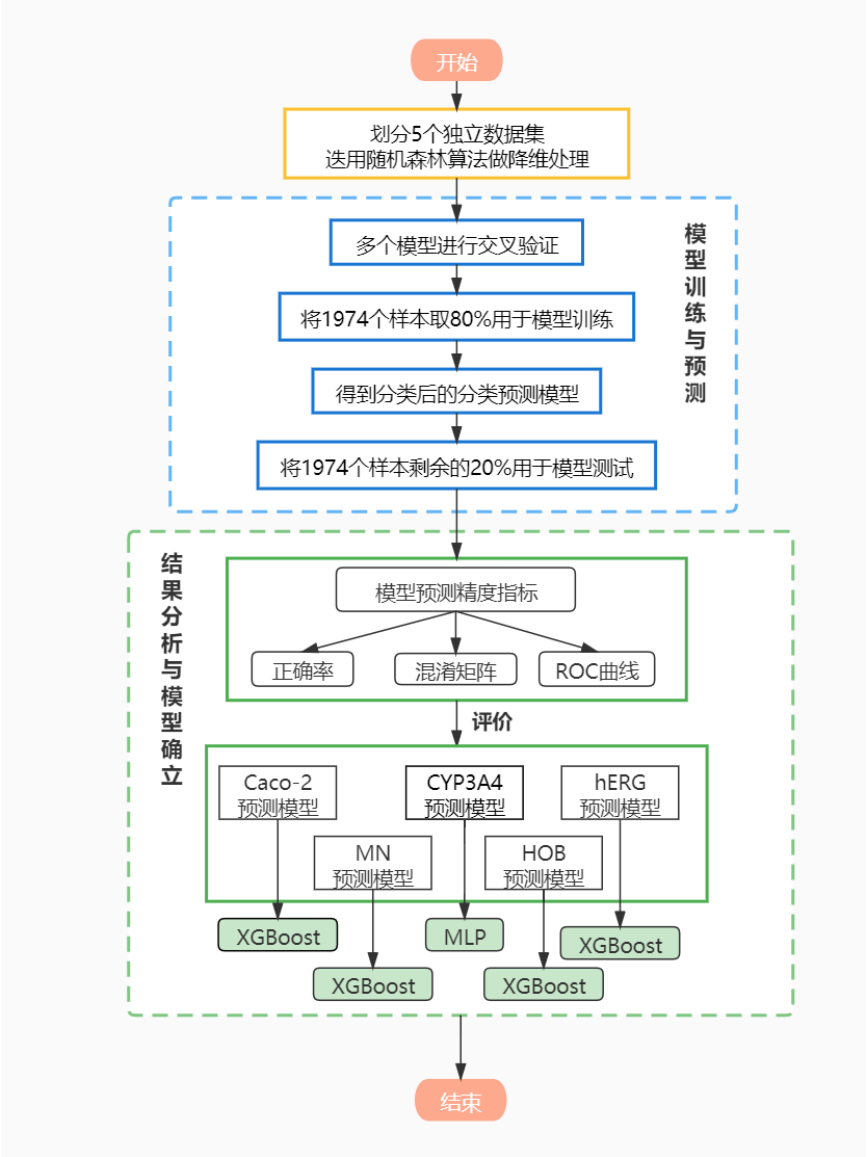


图 6-1 问题三思路流程图

6.2 数据处理

由于题目考虑的 ADMET 性质分别为 Caco-2、CYP3A4、hERG、HOB、MN，因此将该五个性质对应到分子描述符文件，得到 5 个独立数据集。分别对 5 个 ADMET 单独进行分析并构建分类预测模型。由于第一问已经对分子描述符相关数据进行了预处理，本问直接使用处理后的分子描述符数据，考虑问题一处理后分子描述符变量还有 504 个，若不做降维处理，很有可能出现“维数灾难”，因此有必要对变量进行降维。考虑使用随机森林算法分别对 Caco-2、CYP3A4、hERG、HOB、MN 相应的数据进行特征选择，筛选出各自的 50 个主要特征作为下一步模型训练的输入变量。

6.3 ADMET 分类预测模型的建立

本问分别对 5 个独立的性质建立分类预测模型。根据数据特征，主要使用了 9 种机器学习算法（AdaBoost、KNN、DT、LR、GBDT、XGBoost、SVM、RF、MLP）。由于篇幅有限，本文不再对这 9 个模型一一介绍。为了针对每个 ADMET 性质能够选择使用一个最优的分类模型,同时也为验证上述 9 种机器学习模型的稳定性(模型在数据上的泛化能力)，需要确保由数据集得到的分类预测模型已经获得数据集大部分正确的信息，且不包含太多噪声，也就是说，使模型的偏差和方差是较小的。本文通过十折交叉验证来评估 9 个模型在独立数据集上的概括能力，结果如表所示。

表 6-1 9 个机器学习模型在 Caco-2 中的十折交叉验证正确率

Num	AdaBoost	KNN	Decision Tree	Logistic	GBDT	XGBoost	SVM	Random Forest	MLP
1	0.889	0.885	0.853	0.857	0.907	0.922	0.895	0.927	0.902
2	0.87	0.872	0.843	0.838	0.899	0.899	0.875	0.884	0.872
3	0.87	0.853	0.857	0.838	0.89	0.897	0.841	0.897	0.884
4	0.83	0.86	0.815	0.838	0.862	0.865	0.843	0.867	0.87
5	0.862	0.865	0.884	0.85	0.882	0.899	0.867	0.901	0.899
6	0.863	0.818	0.845	0.848	0.887	0.882	0.879	0.884	0.884
7	0.85	0.847	0.835	0.858	0.88	0.887	0.865	0.877	0.877
8	0.857	0.84	0.835	0.86	0.89	0.899	0.87	0.885	0.872
9	0.862	0.855	0.85	0.85	0.874	0.89	0.87	0.894	0.877
10	0.87	0.862	0.847	0.857	0.899	0.902	0.885	0.902	0.901

表 6-2 9 个机器学习模型在 CYP3A4 中的十折交叉验证正确率

Num	AdaBoost	KNN	Decision Tree	Logistic	GBDT	XGBoost	SVM	Random Forest	MLP
1	0.938	0.902	0.895	0.929	0.943	0.936	0.924	0.934	0.922
2	0.921	0.914	0.884	0.924	0.929	0.949	0.924	0.946	0.941
3	0.922	0.927	0.884	0.921	0.924	0.929	0.921	0.927	0.941
4	0.914	0.901	0.904	0.926	0.914	0.924	0.933	0.917	0.917
5	0.938	0.929	0.921	0.946	0.944	0.941	0.929	0.936	0.944
6	0.934	0.916	0.89	0.927	0.929	0.934	0.934	0.938	0.944
7	0.924	0.917	0.902	0.926	0.933	0.939	0.922	0.927	0.938
8	0.926	0.922	0.904	0.922	0.931	0.936	0.929	0.934	0.934
9	0.912	0.916	0.902	0.922	0.919	0.934	0.926	0.931	0.938
10	0.922	0.907	0.895	0.906	0.921	0.929	0.914	0.917	0.938

表 6-3 9 个机器学习模型在 hERG 中的十折交叉验证正确率

Num	AdaBoost	KNN	Decision Tree	Logistic	GBDT	XGBoost	SVM	Random Forest	MLP
1	0.836	0.848	0.835	0.836	0.872	0.885	0.848	0.877	0.884
2	0.841	0.858	0.847	0.838	0.877	0.892	0.85	0.867	0.87
3	0.86	0.872	0.852	0.831	0.875	0.889	0.87	0.879	0.877
4	0.835	0.867	0.857	0.847	0.892	0.901	0.88	0.897	0.882
5	0.863	0.858	0.853	0.848	0.887	0.887	0.875	0.899	0.875
6	0.833	0.836	0.838	0.843	0.868	0.879	0.845	0.877	0.872
7	0.831	0.843	0.813	0.826	0.867	0.862	0.843	0.867	0.865
8	0.85	0.845	0.823	0.847	0.86	0.875	0.85	0.879	0.879
9	0.84	0.857	0.843	0.84	0.879	0.904	0.884	0.892	0.895
10	0.847	0.848	0.852	0.855	0.882	0.902	0.858	0.895	0.897

表 6-4 9 个机器学习模型在 HOB 中的十折交叉验证正确率

Num	AdaBoost	KNN	Decision Tree	Logistic	GBDT	XGBoost	SVM	Random Forest	MLP
1	0.83	0.798	0.821	0.823	0.857	0.872	0.84	0.872	0.86
2	0.815	0.789	0.793	0.777	0.84	0.847	0.815	0.858	0.823
3	0.82	0.782	0.801	0.793	0.831	0.848	0.808	0.843	0.836
4	0.809	0.786	0.799	0.784	0.835	0.84	0.786	0.835	0.838
5	0.816	0.808	0.83	0.821	0.857	0.847	0.815	0.845	0.847
6	0.816	0.799	0.801	0.793	0.838	0.86	0.799	0.853	0.833
7	0.823	0.811	0.808	0.794	0.853	0.857	0.808	0.858	0.841
8	0.841	0.811	0.841	0.82	0.858	0.87	0.82	0.858	0.845
9	0.826	0.825	0.788	0.815	0.847	0.862	0.815	0.838	0.848
10	0.821	0.776	0.82	0.799	0.84	0.858	0.796	0.847	0.833

表 6-5 9 个机器学习模型在 MN 中的十折交叉验证正确率

Num	AdaBoost	KNN	Decision Tree	Logistic	GBDT	XGBoost	SVM	Random Forest	MLP
1	0.906	0.912	0.931	0.889	0.926	0.951	0.894	0.953	0.927
2	0.927	0.894	0.926	0.884	0.938	0.943	0.892	0.941	0.938
3	0.929	0.894	0.897	0.884	0.944	0.953	0.901	0.944	0.944
4	0.912	0.914	0.921	0.895	0.938	0.943	0.897	0.948	0.938
5	0.899	0.89	0.921	0.894	0.938	0.965	0.899	0.961	0.927
6	0.916	0.882	0.922	0.895	0.931	0.941	0.889	0.951	0.927
7	0.907	0.899	0.904	0.892	0.934	0.948	0.868	0.946	0.944
8	0.892	0.897	0.894	0.865	0.914	0.912	0.897	0.919	0.916
9	0.926	0.894	0.921	0.907	0.934	0.948	0.895	0.931	0.931
10	0.916	0.894	0.917	0.895	0.943	0.949	0.919	0.939	0.944

箱线图利用数据的五个统计量可以识别数据是否具有对称性、分布离散情况等信息。因此为了便于观察，对交叉验证得到的正确率绘制箱线图，结果如图所示。

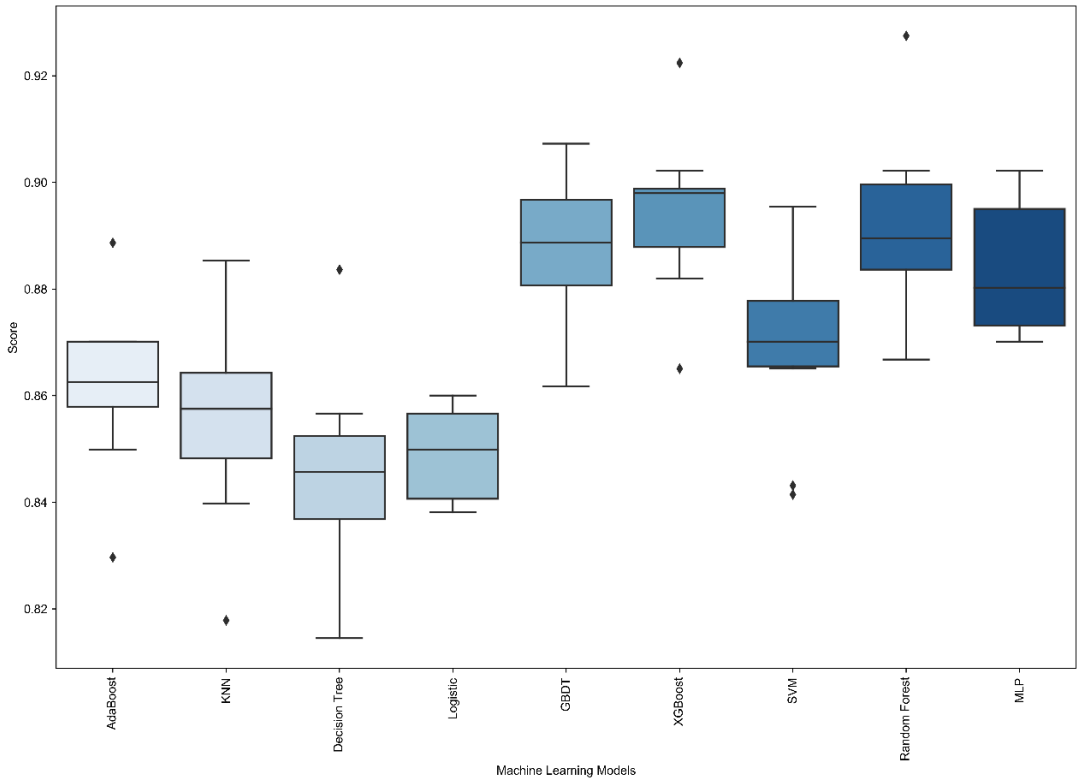


图 6-2 9 个机器学习模型在 Caco-2 中的十折交叉验证正确率的箱线图

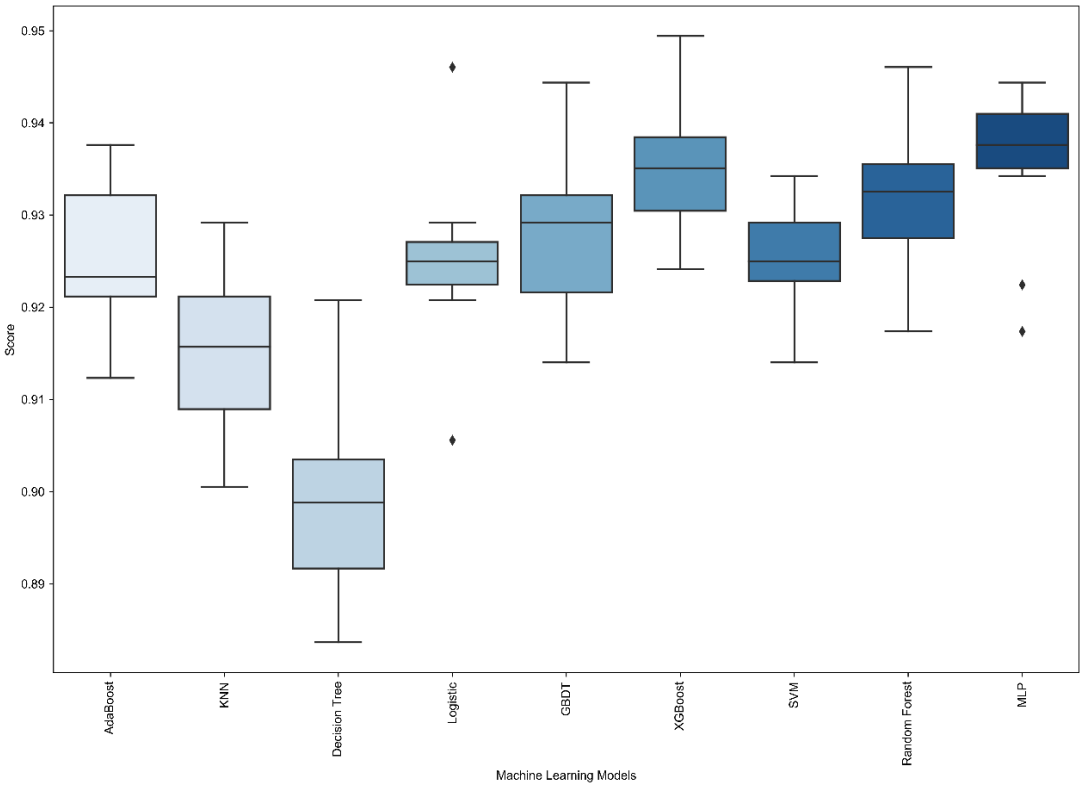


图 6-3 9 个机器学习模型在 CYP3A4 中的十折交叉验证正确率的箱线图

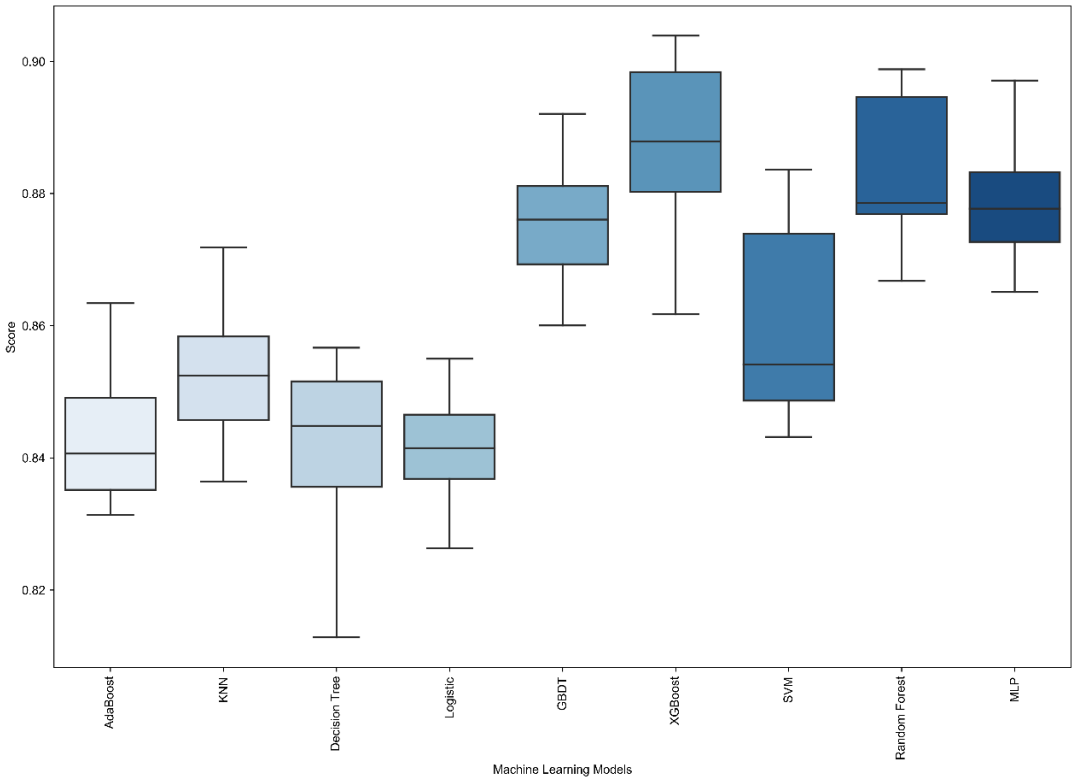


图 6-4 9 个机器学习模型在 hERG 中的十折交叉验证正确率的箱线图

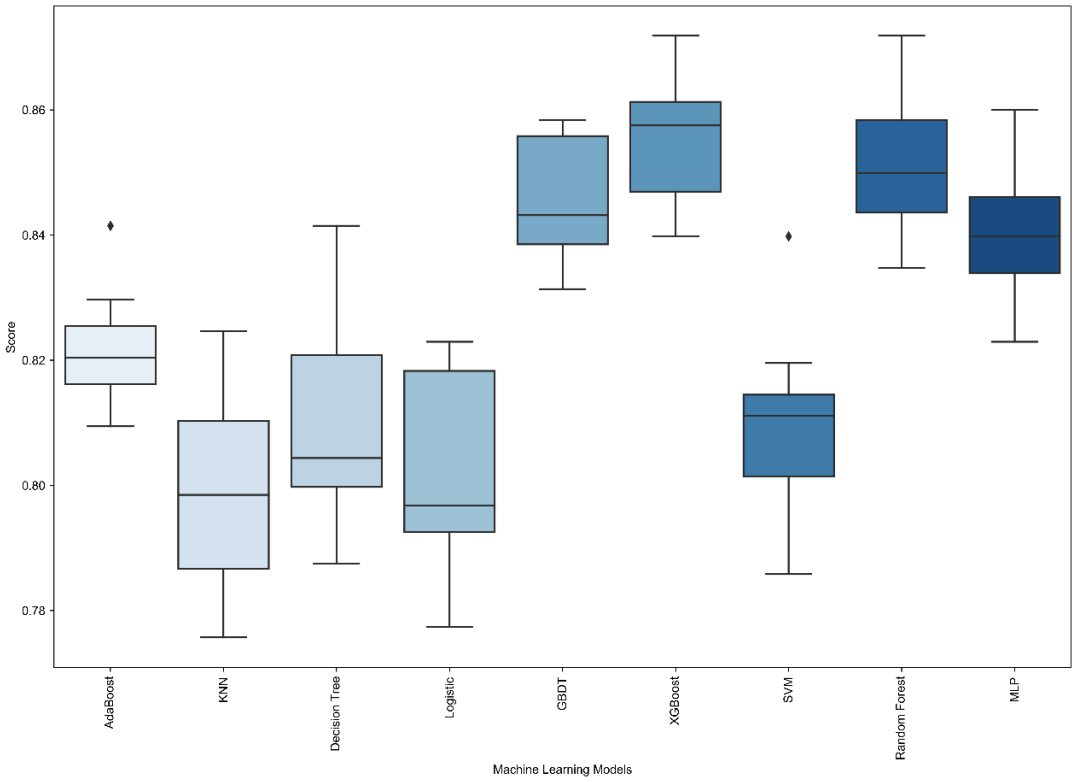


图 6-5 9 个机器学习模型在 HOB 中的十折交叉验证正确率的箱线图

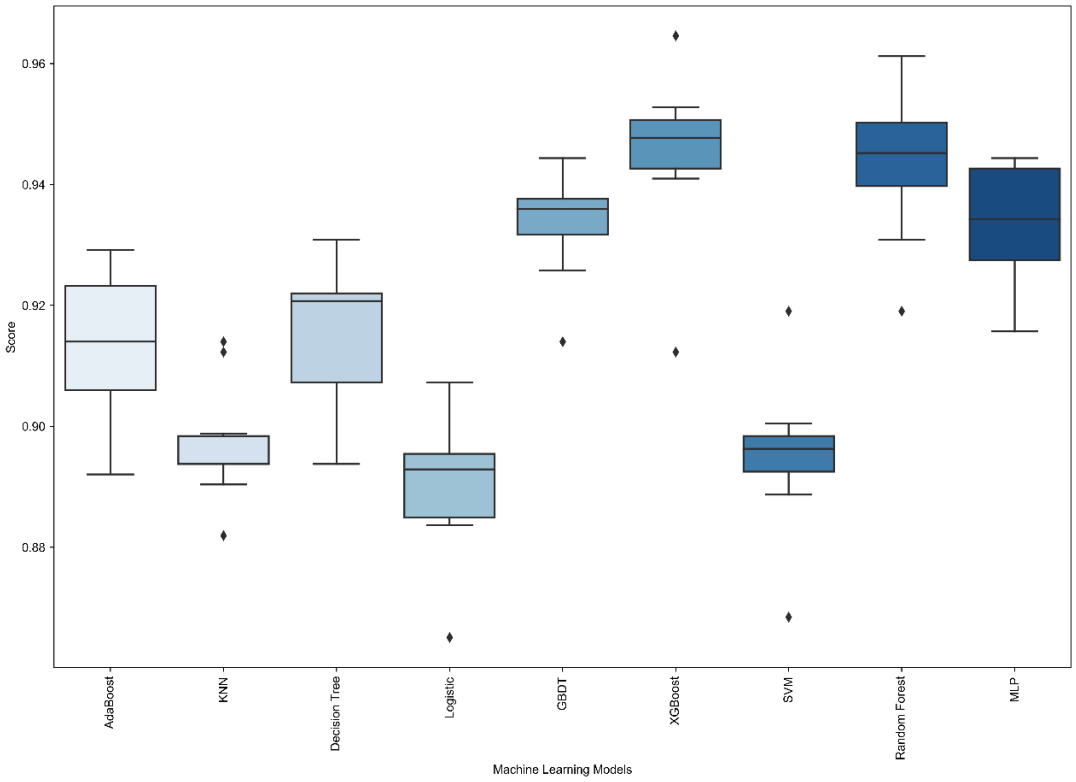


图 6-6 9 个机器学习模型在 MN 中的十折交叉验证正确率的箱线图

从箱线图中可以发现，交叉验证结果中平均（中位数）预测正确率低于 0.88 的模型分别为：AdaBoost、KNN、决策树、逻辑回归。 SVM 的平均（中位数）预测正确率低于 MLP，除此之外，其他四个模型的正确率差异不大。平均（中位数）预测正确率最高的三个模型分别是 XGBoost、随机森林、GBDT。从离散程度来看，XGBoost 离散程度较小，GBDT 与随机森林离散程度相对大一些，XGBoost 与随机森林均存在离群值。综合考虑模型预测精度、离群点、稳定性，Caco-2 较为理想的模型是 XGBoost、GBDT、随机森林, CYP3A4 较为理想的模型是 MLP、XGBoost、RF，hERG 较为理想的模型是 XGBoost、RF，HOB 较为理想的模型是 XGBoost、GBDT、RF，MN 较为理想的模型是 XGBoost、GBDT、RF、MLP。

6.4 九种算法对比

为了进一步确定并验证每个数据集的最优模型，使用 9 种分类预测模型算法进一步对该分类问题预测，并采用以下指标直接或间接地衡量预测精度：正确率、混淆矩阵、ROC 曲线下面积，从而得到 9 种模型对不同 ADMET 指标的分类预测差异，以此来选择各性质所对应的最佳模型。

6.4.1 混淆矩阵

热力图体现了两个离散变量之间的组合关系，通过热力图我们可以非常直观地观察数据大小的差异状况。因此本文将 9 种模型预测结果的混淆矩阵通过热力图呈现，如图所示。

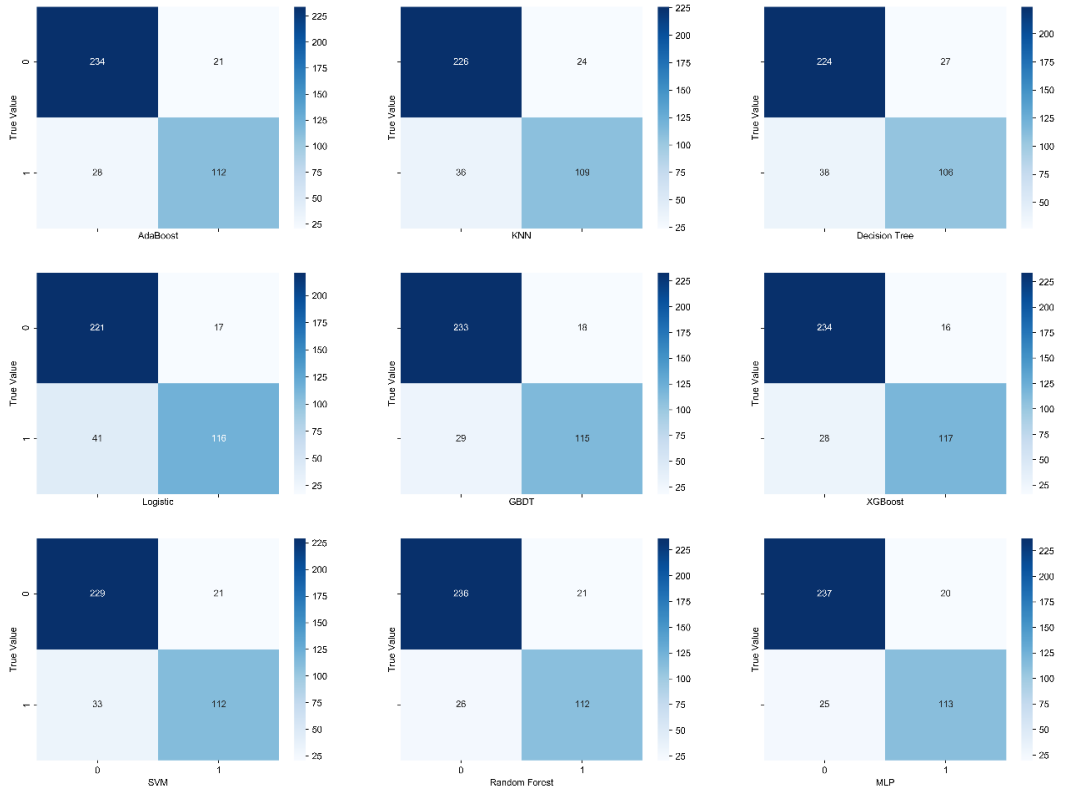


图 6-7 9 种机器学习模型在 Caco-2 中的预测混淆矩阵的热力图

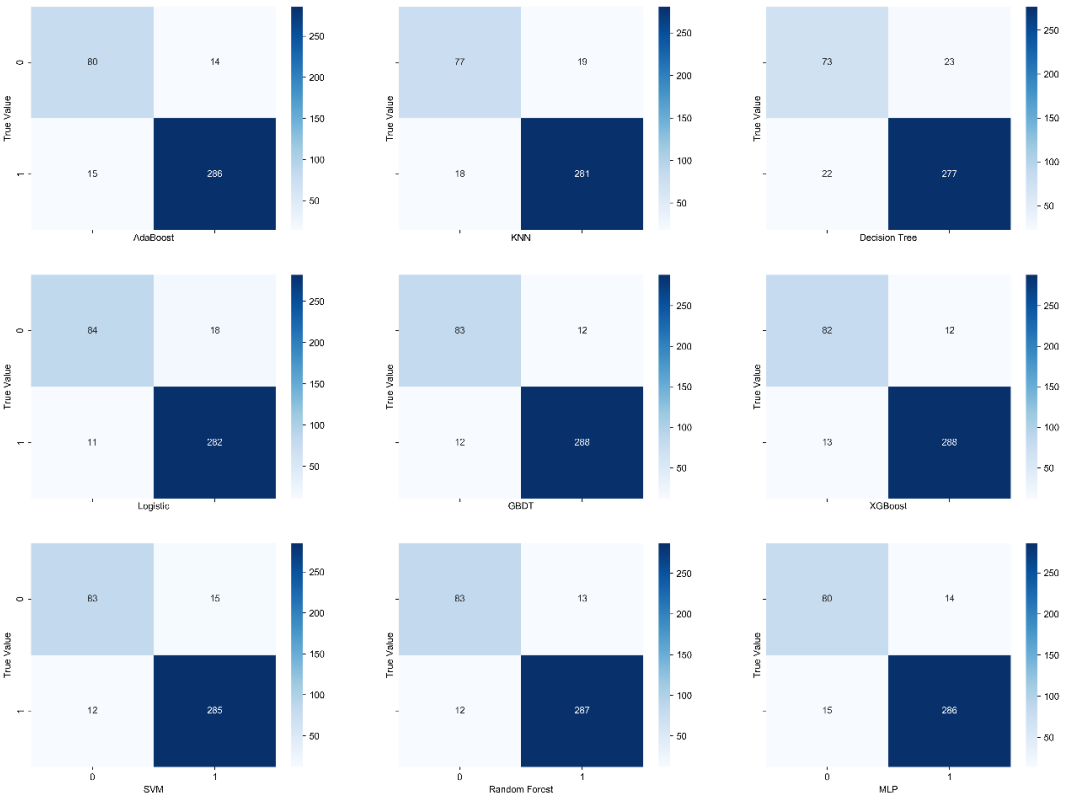


图 6-8 9 种机器学习模型在 CYP3A4 中的预测混淆矩阵的热力图

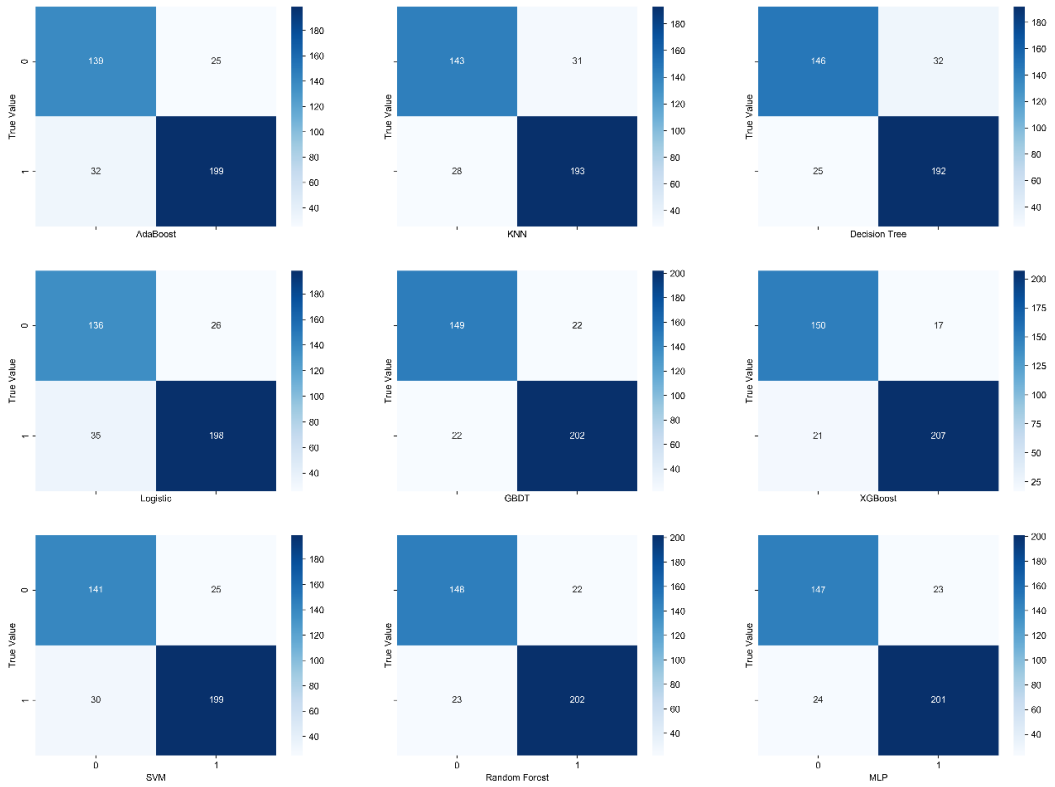


图 6-9 9 种机器学习模型在 hERG 中的预测混淆矩阵的热力图

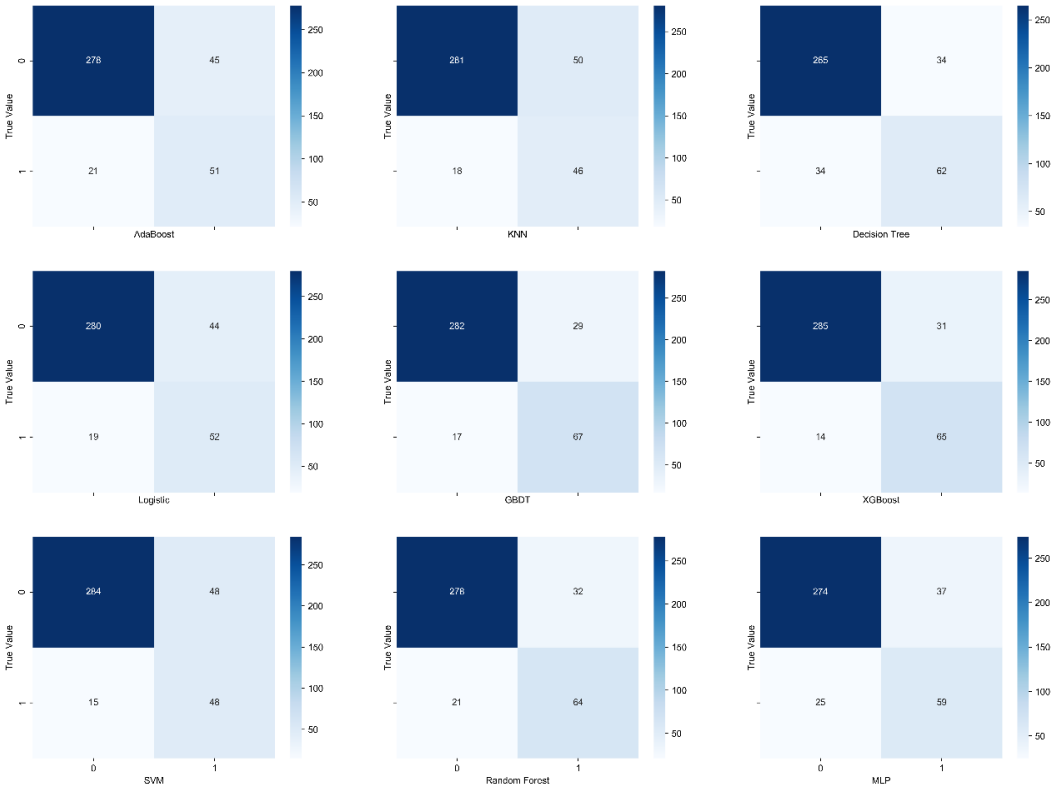


图 6-10 9 种机器学习模型在 HOB 中的预测混淆矩阵的热力图

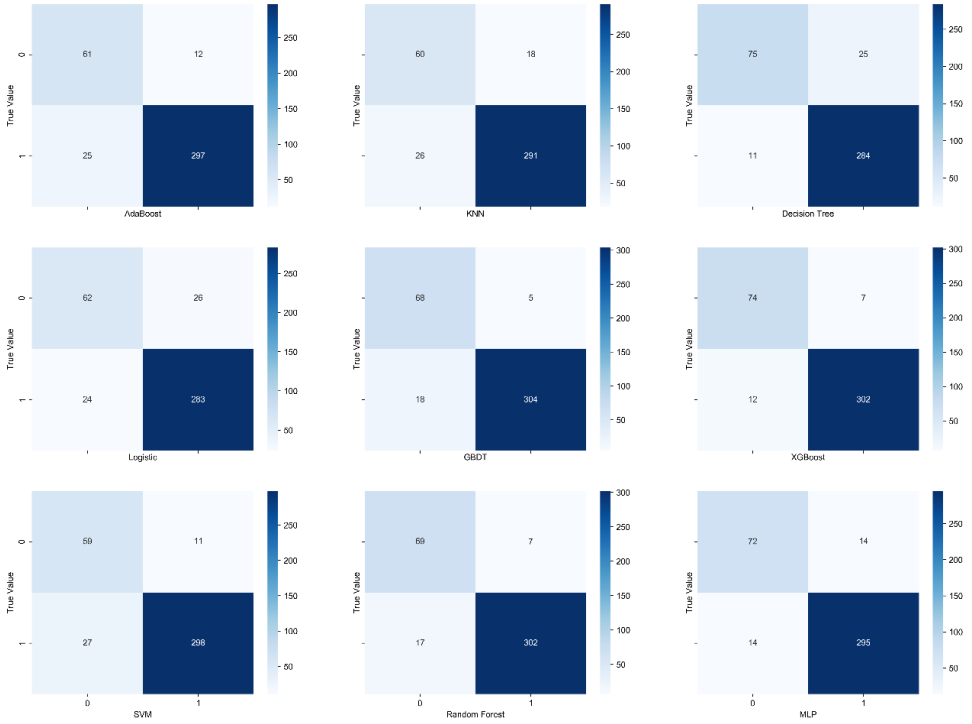


图 6-11 9 种机器学习模型在 MN 中的预测混淆矩阵的热力图

从图中可以发现，对于正确预测识别真假信息的比例在 9 种模型都较高，但是在 Caco-2 中，KNN、DT、LR、SVM 更易预测出错误的结果；CYP3A4 中，AdaBoost、KNN、DT 预测结果错误率较高；hERG 和 HOB 中模型预测错误率普遍偏高，但 GBDT、XGBoost、RF 的错误率相对较低；MN 中 GBDT、XGBoost、RF、MLP 预测错误率较低。

6.4.2 ROC 曲线

对于二分类预测问题，不同的阈值所对应的预测结果和精度都不一样，由于高 TPR 值总是对应着高 FPR 值，所以本文用 ROC 曲线来进行全面的度量比较 9 种模型，如下图所示。

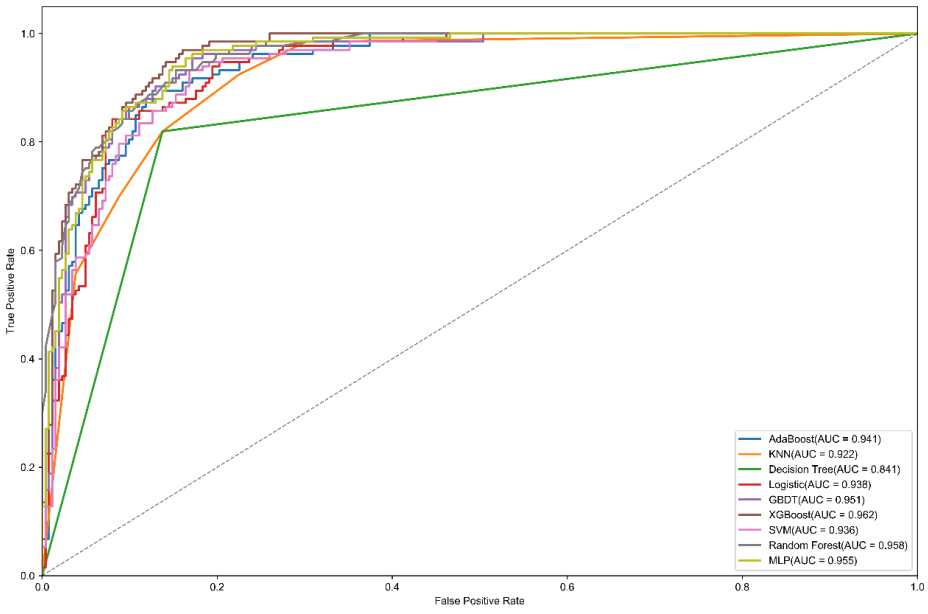


图 6-12 9 种机器学习模型在 Caco-2 中预测的 ROC 曲线

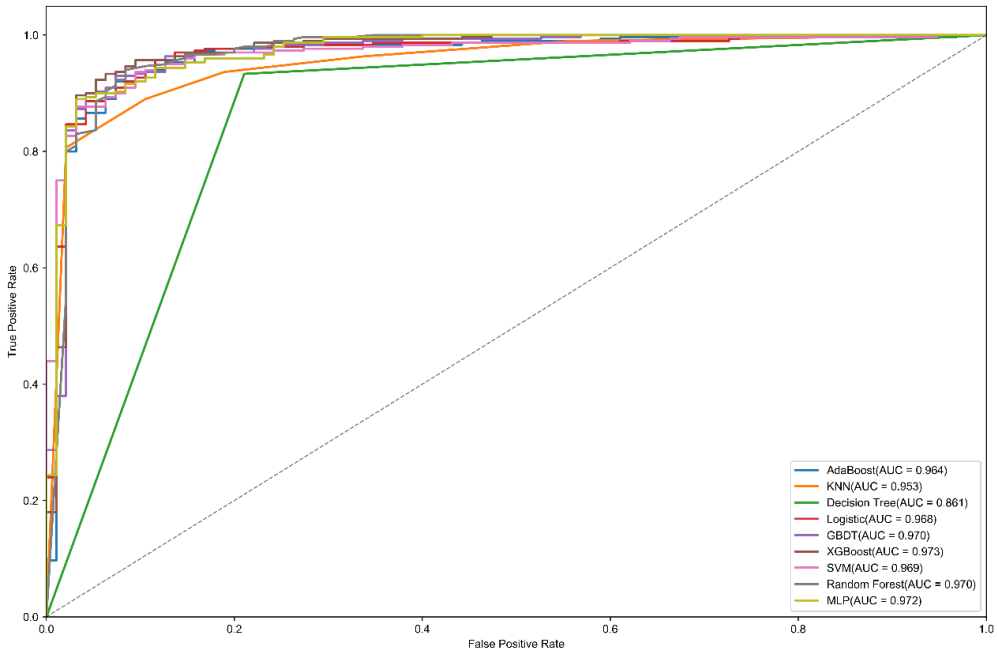


图 6-13 9 种机器学习模型在 CYP3A4 中预测的 ROC 曲线

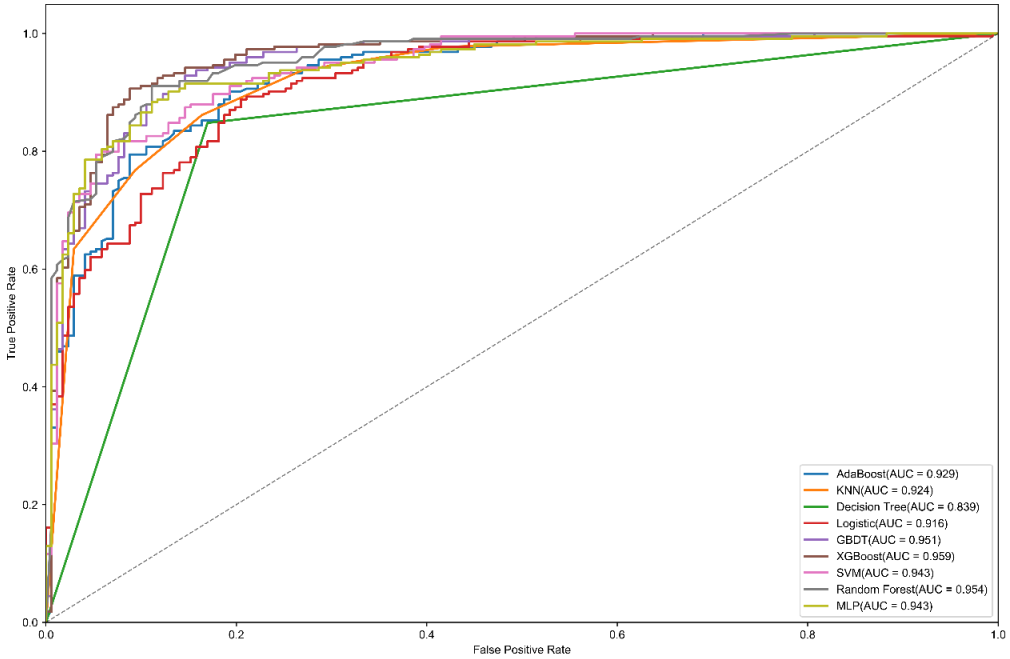


图 6-14 9 种机器学习模型在 hERG 中预测的 ROC 曲线

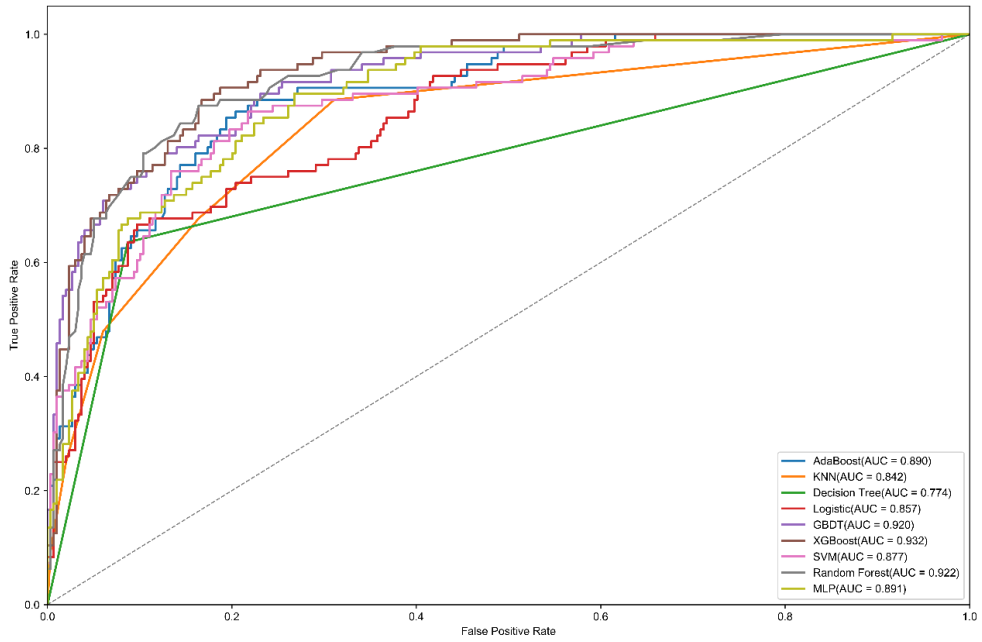


图 6-15 9 种机器学习模型在 HOB 中预测的 ROC 曲线

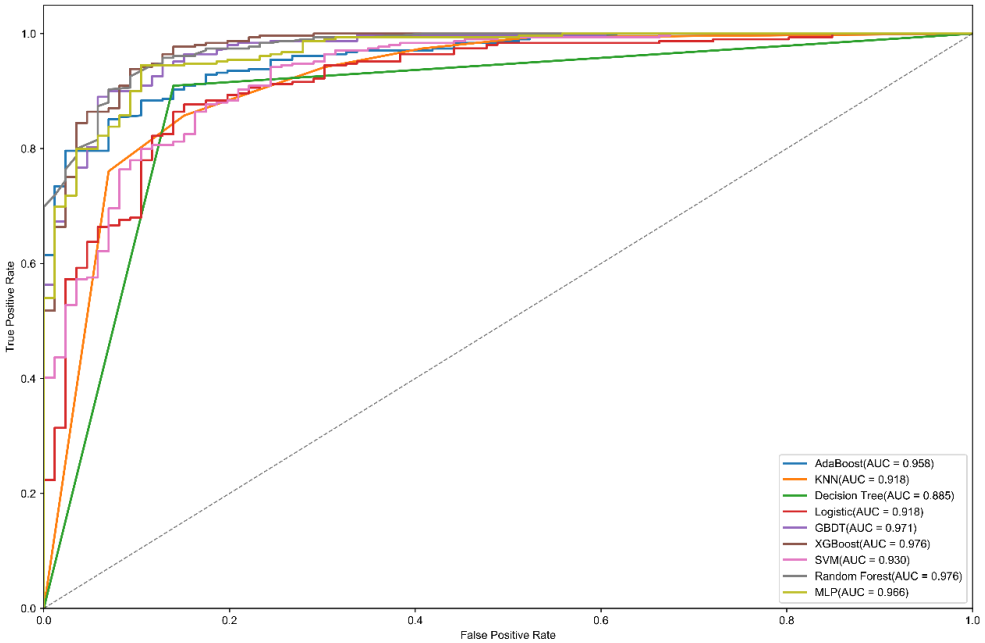


图 6-16 9 种机器学习模型在 MN 中预测的 ROC 曲线

结果显示，ROC 曲线与对角线相比，永远是向上凸起的。这表明我们的 9 种预测模型都是有效的。

Caco-2 中，曲线下面积（AUC）最大的是 FR 模型（AUC=0.957）、XGBoost 模型（AUC=0.956），AUC 最小的两个模型分别是 DT 模型（AUC=0.828）、KNN 模型（AUC=0.922）。由于 DT 模型和 KNN 模型的 AUC 值较低，因此可以得到 DT 和 KNN 模型的预测精度较差。而其他模型的 AUC 曲线非常相似，也就是说他们的预测能力相当。

CYP3A4 中，曲线下面积（AUC）最大的是 XGBoost 模型（AUC=0.973）、MLP 模型（AUC=0.972），AUC 最小的模型是 DT 模型（AUC=0.861）。由于 DT 模型和 KNN 模型

的 AUC 值较低，因此可以得到 DT 模型的预测精度较差。

hERG 中，曲线下面积（AUC）最大的是 XGBoost 模型（AUC=0.959）、RF 模型（AUC=0.954），AUC 最小的两个模型分别是 DT 模型（AUC=0.839）、LR 模型（AUC=0.916）。由于 DT 模型和 LR 模型的 AUC 值较低，因此可以得到 DT 和 LR 模型的预测精度较差。

HOB 中，曲线下面积（AUC）最大的是 XGBoost 模型（AUC=0.931）、RF 模型（AUC=0.922），AUC 最小的两个模型分别是 DT 模型（AUC=0.774）、KNN 模型（AUC=0.842）。由于 DT 模型和 KNN 模型的 AUC 值较低，因此可以得到 DT 和 KNN 模型的预测精度较差。

MN 中，曲线下面积（AUC）最大的是 XGBoost 模型（AUC=0.976）、RF 模型（AUC=0.976），GBDT 模型（AUC=0.971），AUC 最小的两个模型分别是 DT 模型（AUC=0.885）、LR 模型（AUC=0.918）。由于 DT 模型和 LR 模型的 AUC 值较低，因此可以得到 DT 和 LR 模型的预测精度较差。

综上，在所有模型预测结果的 AUC 值表现中，XGBoost 与 RF 表现总是最好的，预测精度较高，而 DT、KNN、LR 三个模型预测精度较差。

6.5 应用模型预测

综合考虑 9 种机器学习模型预测结果的混淆矩阵、ROC 曲线线下面积、正确率，分别对 5 种 ADMET 指标选出最佳分类预测模型如下表 6-6 所示。

表 6-6 化合物其他 ADMET 指标的分类预测模型

化合物 ADMET 指标	预测模型
Caco-2	XGBoost
CYP3A4	MLP
hERG	XGBoost
HOB	XGBoost
MN	XGBoost

XGBoost 所选用的预测模型主要参数设置如下表所示：

表 6-7 XGBoost 主要参数设置

符号	含义	取值
eval_metric	评估方法	['logloss', 'auc', 'error']
learning_rate	学习率	0.300000012
max_depth	给定树的深度	6
n_estimators	子模型数量	100
n_jobs	线性并行个数	8

表 6-8 MLP 主要参数设置

符号	含义	取值
Hidden_layer_sizes	隐藏层神经元个数	(100,)
alpha	正则化项参数	0．0001
learning_rate	学习率	constant
max_iter	最大迭代次数	200
tol	优化容忍度	0.0001

分别对待测数据进行预测，预测结果如表 6-9 所示。

表 6-9 ADMET 预测模型用在 test 表数据的预测结果

No	Caco-2	CYP3A4	hERG	HOB	MN
1	1	1	1	0	1
2	1	0	0	1	1
3	1	0	1	1	1
4	1	0	0	1	1
5	1	0	0	1	1
6	1	0	0	1	1
7	1	0	0	0	1
8	1	0	0	1	1
9	1	0	0	1	1
10	1	0	1	1	1
11	1	0	1	1	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	0	1
16	1	1	1	0	1
17	1	0	1	1	1
18	1	1	0	1	1
19	1	0	0	1	1
20	1	0	0	1	1
21	1	0	1	0	1
22	1	1	0	1	1
23	1	0	0	1	0
24	1	0	0	1	0
25	1	1	0	1	1
26	1	1	0	1	1
27	1	0	1	1	1
28	1	1	1	1	1
29	1	1	1	1	1
30	1	1	1	1	1
31	1	1	1	0	1
32	1	1	1	0	1
33	1	1	1	1	1
34	1	1	1	0	1
35	0	1	1	0	1
36	0	1	1	0	0
37	0	1	1	0	1
38	1	1	1	0	0
39	0	1	0	0	1
40	0	1	0	0	1
41	0	1	0	0	1

42	0	1	0	0	1
43	0	1	0	0	1
44	0	1	0	0	1
45	0	1	0	0	1
46	1	1	0	1	1
47	1	1	1	1	1
48	1	1	0	1	1
49	1	1	1	1	1
50	1	1	1	1	1

七、问题四：化合物的分子描述符的选取及取值

7.1 问题四分析

本题要求寻找具有更好生物活性的分子描述符，并求得这些分子描述符的最优取值或最优取值范围，同时这些分子描述符要具有更好的 ADMET 性质。根据上述分析，可以将问题简化为基于约束条件的目标优化问题，则优化目标是最优化化合物的生物活性，约束条件包含具有更好的 ADMET 性质在内的多条约束。在该问中为提高化合物的生物活性，考虑选取 pIC_{50} 作为优化目标，因为 pIC_{50} 为 IC_{50} 值的负对数，而 pIC_{50} 的值越大表面生物活性越高，且在实际的 QSAR 建模中，往往采用 pIC_{50} 表示生物活性值。本题的思路流程如图 7-1 所示：

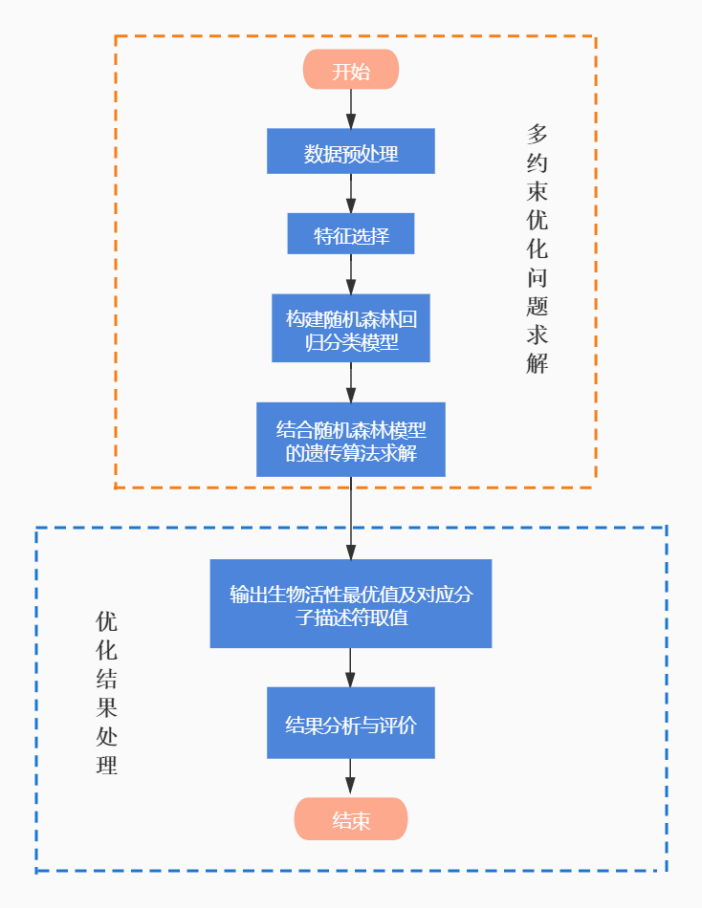


图 7-1 问题四思路流程图

本问的难点在于：

（1）需要从 729 个分子描述符中找到符合要求的描述符作为优化的决策变量，其中描述符要对于生物活性值及 5 个 ADMET 性质均具有显著影响。

（2）需要依据所选取具有显著影响的主要决策变量，分别构建对于生物活性值和 5 个 ADMET 性质的回归预测模型，将其作为目标函数以及约束函数。

（3）在满足约束条件下合理地对化合物的生物活性进行优化处理，求解其对应操作变量的取值或取值范围。

（4）特别注意 ADMET 性质中 hERG 和 MN 的取值为 1 时，表示该化合物具有心脏毒性和遗传毒性，需要进行特殊处理。

考虑到问题本身的复杂性以及分子描述符样本的高维度，该问采用问题一数据预处理后的数据，随后需要对其进行依据 pIC_{50} 以及 5 个 ADMET 性质分别作特征选择操作，选取重要性排名前 20 位的分子描述符作为主要决策变量。

通过对第二题与第三题的回归预测结果分析，随机森林回归预测和分类模型对于 pIC_{50} 以及 5 个 ADMET 性质的预测和分类效果较优。因此在本问中，继续使用随机森林算法分别对 pIC_{50} 以及 5 个 ADMET 性质构建回归及分类模型，最后结合优化算法进行优化。

针对本问的优化模型，为便于与回归预测模型相结合，本文选用具有内在隐并行性和更好全局寻优能力的遗传算法进行求解。

7.2 数据预处理及特征选择

采用问题一数据预处理后的结果并使用 8 种特征选取方法，分别针对 pIC_{50} 以及 5 个 ADMET 性质进行特征选择，分别得到 8 个方法对 pIC_{50} 以及 5 个 ADMET 性质的分子描述符重要性排序，其中对于 pIC_{50} 的分子描述符重要性排序如图 7-2 所示（对于 5 个 ADMET 性质的分子描述符重要性排序图见附录）。

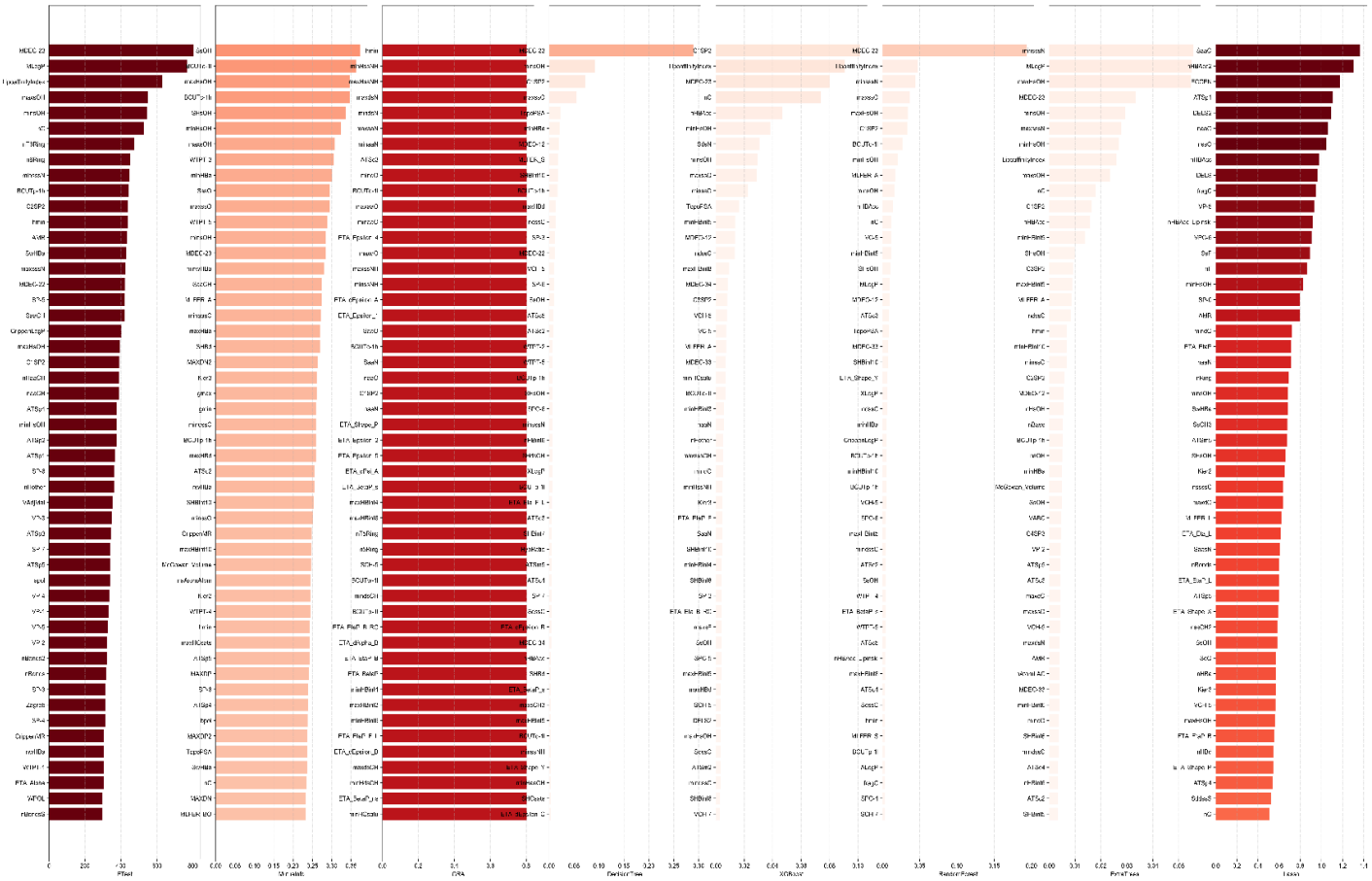


图 7-2 pIC_{50} 的分子描述符重要性排序

对于 6 个指定指标（pIC_{50} 以及 5 个 ADMET 性质），统计 8 种方法前 50 个变量出现的总频率，选取排名靠前的 20 个变量，再综合选取 6 个指定指标中共同出现频率最高的 20 个分子描述符作为特征选择的结果。

最终选取的 20 个分子描述符如下表 7-1 所示：

表 7-1 分子描述符选取结果

20 个分子描述符选取结果展示			
1	minHBa	11	TopoPSA
2	MDEC-23	12	AMR
3	SP-1	13	BCUTc-1l
4	maxHsOH	14	VABC
5	WTPT-1	15	minsOH
6	maxsCH3	16	MLFER_E
7	VP-1	17	CrippenMR
8	MLFER_A	18	VP-0
9	minsssN	19	SP-2
10	LipoaffinityIndex	20	minHsOH

7.3 模型建立

7.3.1 数学模型

（1）决策变量

该问题的样本数据经过前一小节降维后得到了对 pIc_{50}、Caco-2、CYP3A4、hERG、HOB、MN 都具有较大影响的 20 个主要变量，将这 20 个主要变量作为决策变量输入到本题所建立的优化模型中，记作：

$$X=\{X_1,X_2,X_3,...,X_{20}\}.$$

（2）目标函数

本题要求根据所寻找的主要分子描述符，求出当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。

首先为提高化合物对抑制 *ERα* 的生物活性，考虑选取 pIC_{50} 作为优化目标，因为 pIC_{50} 为 IC_{50} 值的负对数，pIC_{50} 的值越大表面生物活性越高，且在实际的 QSAR 建模中，往往采用 pIC_{50} 表示生物活性值[7]。由于该优化模型中的 pIC_{50} 值由所构建的随机森林回归预测模型得到，因此提出该模型的目标函数为：

$$\max pIC_{50}=f_{pIC_{50}}(X)$$

（3）约束条件

本题要求在使化合物对抑制 *ERα* 具有更好的生物活性的同时，要具有更好的 ADMET 性质，即要求对于 Caco-2、CYP3A4、hERG、HOB、MN 五个性质至少有 3 个为优，特别注意的是其中 hERG 和 MN 取值为 1 时表示该化合物具有心脏毒性和遗传毒性，而 Caco-2、CYP3A4、HOB 取值为 1 时的性质则为优。因此为计算简便，将 hERG 与 MN 的取值含义取反，即取值为 1 时代表性质较好，取值为 0 时代表性质较差（此含义仅在算法计算中修改，不影响输入和输出数据）。该约束表示为：

$$\sum_{i=1}^{5} z_i^{ADMET} \geq 3$$

而为了确保各分子描述符取值的合理性，假定各分子描述符的取值范围是其最大最小值范围，约束表示如下：

$$x_i \in (\min(X_i), \max(X_i)), i = 1, 2, ..., 1974$$

$X_i = \{x_1, x_2, x_s ..., x_{1974}\}, i = 1, 2, ..., 20$表示第$i$类分子描述符的取值集合。

（4）模型参数设定

针对该问的要求结合相关因素，选取遗传算法主要参数设定如表 7-2 所示：

表 7-2 遗传算法主要参数设定

符号	含义	取值
m	种群规模	60
P	交叉概率	0.9
Q	变异概率	0.05
Iter	迭代次数	800

7.3.2 基于随机森林预测的遗传算法优化模型

由于该问题的目标函数及个别约束条件是由随机森林预测建立的一个黑箱模型，具体的目标函数表达式无法得知，因此常规的优化算法无法进行有效求解，但是遗传算法作为一种启发式算法，在优化求解过程中无需输入确切的函数表达式，只需将具体数值输入算法便可迭代优化求解，而且相比于其他算法，遗传算法更容易与随机森林预测模型相结合，所以本问选择遗传算法进行随机森林预测模型求解[8]。

首先对于 6 个预测目标（pIC_{50} 以及 5 个 ADMET 性质）分别使用随机森林算法构建 6 个预测模型，其算法模型的预测效果已在前两问中详细阐述，在此不多加赘述。随后使用遗传算法对 pIC_{50} 进行优化，在优化过程中将所得到的每一代自变量输入到所构建的 6 个预测模型中，得到适应度值和 5 个 ADMET 约束值，将其作为遗传算法选择操作的依据，算法迭代直至达到最大迭代次数。

本题整体的算法流程图如下图 7-3 所示：

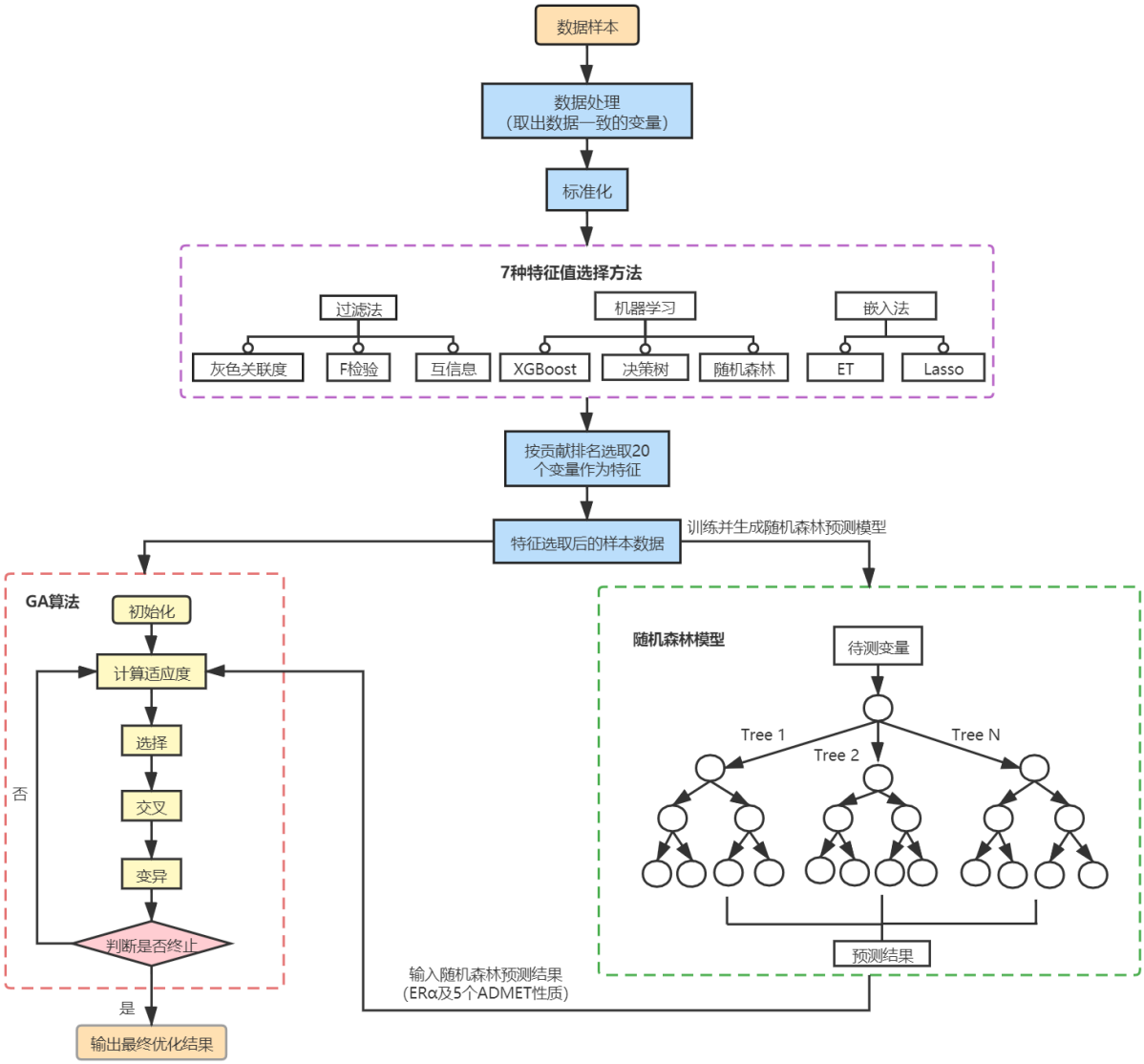


图 7-3 基于遗传算法优化的随机森林预测模型流程图

算法步骤如下：

（1）染色体生成

将每个分子描述符的数值作为染色体的实数编码，并且以每个样本分子描述符的值作为初始染色体水平，以该水平为基准波动生成初始染色体，波动范围为该分子描述符的最大值与最小值区间$[\min(x_i), \max(x_i)]$。初始染色体x_{init}生成规则如下：

$$p = rand(-1,1)$$

$$x_{init} = \begin{cases} x_s + p(\max(X_i) - x_s), p \geq 0 \\ x_s + p(x_s - \min(X_i)), p < 0 \end{cases}$$

（2）初始种群。按照上述染色体生成规则随机生成初始种群。

（3）适应度值。在该题中遗传算法的适应度值由之前所构建的 6 个随机森林预测模型得出。

$$y_{pIC_{50}} = f_{pIC_{50}}(X)$$

$$y_{Caco-2} = f_{Caco-2}(X)$$

$$y_{CYP3A4} = f_{CYP3A4}(X)$$

$$y_{hERG} = f_{hERG}(X)$$

$$y_{HOB} = f_{HOB}(X)$$

$$y_{MN} = f_{MN}(X)$$

（4）选择操作。采用“轮盘赌”选择策略。结合“精英保留”策略的思想，从上一代所有个体中强制保留最优染色体，并剔出最差染色体。随后基于适应度决定每个个体被选择的概率，从剩余染色体中随机选择个体进入新的种群。重复上述操作，直到新的种群达到原来的种群规模。

（5） 交叉操作。交叉操作选择单点交叉，按一定概率选取两条染色体分别对半分割并交换，从而得到两个不同的子染色体，如图 7-4 所示。

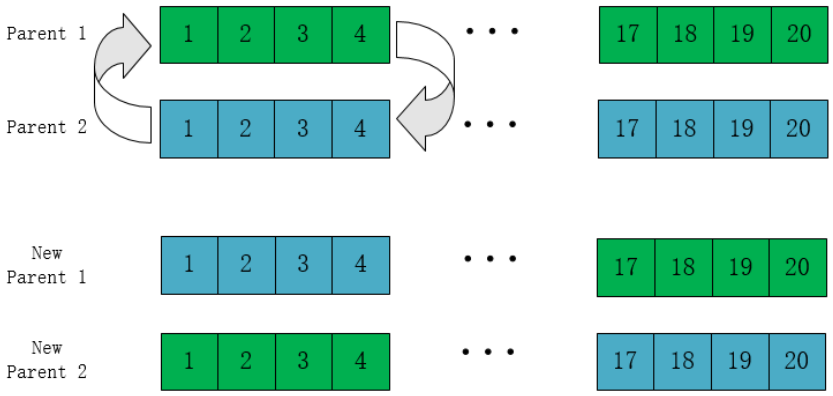


图 7-4 染色体交叉操作

（6） 变异操作。随机选择n个位点的染色体基因在变量的最大最小值范围内进行基因突变得到变异后的子染色体。，突变规则如下：

$$x_i = rand(\min(X_i), \max(X_i))$$

7.4 问题求解与分析

使用 Python 语言根据上述建立的优化模型进行迭代求解，求得优化后的 pIC_{50} 值及其对应的 ADMET 性质，以及 20 个主要分子描述符的取值（保留三位小数）如表 7-3 及 7-4 所示：

表 7-3 pIC_{50} 及其对应 ADMET 性质的取值

名称	数值	含义
pIC50	8.341	对抑制 ERa 具有近似最优的生物活性
Caco-2	1	化合物小肠上皮细胞渗透性较好
CYP3A4	1	化合物能被 CYP3A4 代谢

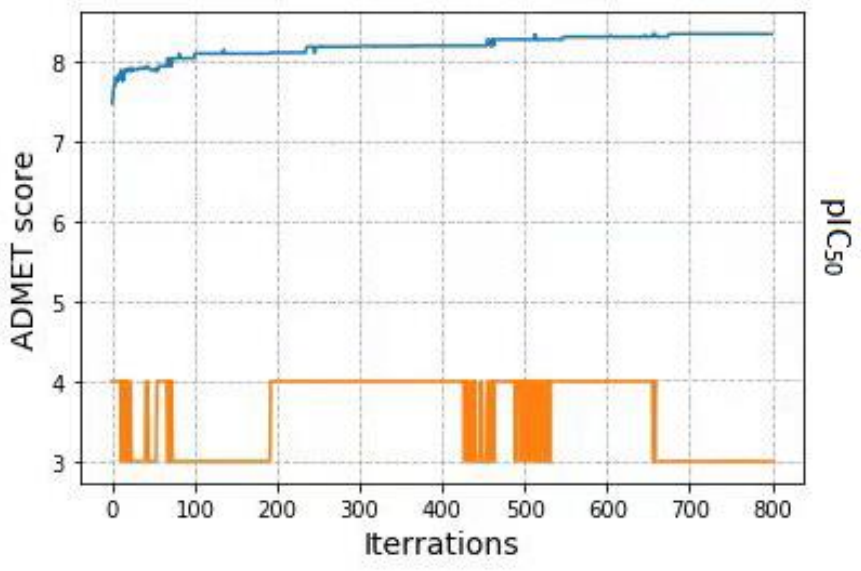
hERG	1	化合物具有心脏毒性
HOB	0	化合物口服生物利用度较差
MN	0	化合物不具有遗传毒性

表 7-4　选取 20 个分子描述符的取值

分子描述符	取值	分子描述符	取值
minHBa	0.780	TopoPSA	27.140
MDEC-23	17.180	AMR	456.687
SP-1	9.863	BCUTc-1l	-0.222
maxHsOH	0.746	VABC	1177.403
WTPT-1	37.244	minsOH	7.075
maxsCH3	0.239	MLFER_E	0.974
VP-1	10.621	CrippenMR	91.205
MLFER_A	5.823	VP-0	17.817
minsssN	1.228	SP-2	13.673
LipoaffinityIndex	22.049	minHsOH	0.684

其中可以看到，当 pIC_{50} 取值为 8.341 达到近似最优，其化合物的 ADMET 性质中 hERG 和 HOB 未达到较优效果，表现为具有心脏毒性及化合物口服生物利用度较差，而其余 3 项性质均达到较优性质，因此该优化结果符合“至少三个性质较好”的条件。分子描述符变量的优化取值也都处于合理的操作范围内。

如图 7-5 所示，为该算法模型的优化迭代收敛图，可以看到算法的 pIC_{50} 值在第 656 次迭代时达到了近似最优值，虽然之后仍有少许波动，但算法的收敛已趋于稳定，而对于 ADMET 性质的评分则由于约束条件的限制，始终在 3 与 4 之间波动（低于 3 的结果均已在遗传算法的选择阶段舍弃），最终在近似最优解处的 ADMET 性质评分为 3。



7-5 迭代收敛图

为了验证模型求解结果的合理性，选取初始数据样本中分子描述符取值与该优化结果的分子描述符取值相近的 5 个样本，以及 5 个 pIC_{50} 值与优化结果相近的样本，进行结果分析比较，如下表所示：

表 7-5　结果分析

样本序号	pIC_{50} 值	Caco-2	CYP3A4	hERG	HOB	MN	ADMET 分值
5	8.409	0	1	1	0	0	2
19	8.155	1	1	1	0	0	3
87	8.174	1	1	1	0	1	2
424	8.097	1	0	0	0	1	2
545	8.476	0	1	1	0	1	1
743	9.215	0	1	1	0	1	1
939	9.000	0	1	1	0	1	1
947	8.398	0	1	1	0	1	1
1434	9.699	0	1	0	0	1	2
1780	8.367	0	1	1	0	0	2
优化值	8.341	1	1	1	0	0	3

分析比较发现，尽管有些样本化合物的 pIC_{50} 值大于该算法的优化值，但其对应的 ADMET 值却无法满足较优的条件，而满足 ADMET 性质的样本的 pIC_{50} 值却较差，因此可以认为该算法的结果达到近似最优，验证了该方法的合理性与准确性。

八、模型检验与评价

8.1 模型优缺点分析

1．模型优点

问题一充分考虑变量之间的非线性关系，进行恰当合理地数据预处理过程，删除数据一致的变量。采用 F 检验、互信息、XGBoost、随机森林等算法计算变量重要性，综合考虑了不同特征选择算法对变量选取的影响，得到重要性排名前 20 的分子描述符。

问题二中通过对问题一得到的前 20 个分子描述符进行距离系数相关分析，剔除 5 个强相关性变量，使用 8 个回归预测模型进行十折交叉验证，再分析比较待预测数据集的预测效果，选定随机森林作为最终的回归预测方法，对待测数据预测能够预测出较好的结果。所建立的随机森林回归预测模型预测结果良好，验证了模型的有效性和稳定性。

问题三选用多个分类预测方法进行交叉验证，得到模型对每个独立数据集的概括能力。训练构建模型后，通过正确率、混淆矩阵、ADMET 线下面积等统计指标分析训练后模型的预测精度，综合分别选取针对 5 个 ADMET 性质指标的最佳分类预测模型。利用该方法得到的预测模型精度高，效果稳定。

问题四将随机森林机器学习算法和遗传算法结合，对化合物活性进行优化。首先该问利用问题一的重要变量选择方法综合选取了共同对生物化合物活性和五个 ADMET 变量重要性排序前 20 的分子描述符作为的主要特征，随后，在该问中巧妙地利用随机森林预测模型替代遗传算法中的适应度函数及 ADMET 约束条件值，用来优化化合物的生物活性。通过算法的迭代进化，最终选取获得了较为优良的优化结果。

考虑药物研究的时间成本和费用，本文得到的化合物活性预测模型和 ADMET 性质的分类预测模型对研究生产抗 雌性激素受体药物有一定参考价值。

2．模型缺点

（1）在建模时认为所有分子描述符都可以独立任意改变，而没有考虑其相互之间的影响。

（2）对重要变量筛选时仅考虑了特征选择方法，并未考虑实际的药物生产过程对变量关系的影响。

（3）样本数据量较少，预测模型的训练效果仍有提升空间。

8.2 模型改进与推广

新药物的研发往往需要大量的时间和生产成本，为提高药物研究效率，有必要对候选药物活性建立恰当的预测模型，因此本文所提出的预测模型和优化方法可推广至其它药物的研发流程。

（1）本文数据分析与预测模型建立均采用多个方法对比，综合取优。

（2）得到的模型适应性强且稳定，对以后的生物药物研究有一定的现实参考意义。

（3）为提高模型预测结果，可考虑深度学习进行预测分析。

问题二、三和四中的模型，分别对化合物的分子描述符进行了优化，对生物活性值进行了预测进行了预测，在题设区间里，验证了模型结果的合理性。

参考文献：

[1]王翔. 性激素与乳腺癌之间的关系[J].抗癌之窗 2019,(04).
[2]孙少康, 黄勇, 李志明, 龙凤, 世界中医药. 生物活性多糖抗乳腺癌作用研究进展[J]. 2021,16(18).
[3]唐初. 基于雌激素受体和组蛋白去乙酰化酶的新型双靶点抗乳腺癌药物的设计、合成及生物活性研究[D]. 武汉大学 2015.
[4]熊佐松, 黄辉, 陈春锋, 孙珊珊, 马萌宏, 史彬, 鄢烈祥. 应用智能可视化优化方法确定催化重整的优化操作区域[J]. 西安交通大学学报，51(03):136-140, 2017.
[5]Zhou Zhihua. Machine Learning [M]. Beijing: Tsinghua University Publishing House,2016:425.
周志华,机器学习[M]. 清华大学出版社，2016.
[6]狄培琰,康乐,孟驿佳,柴彬,苗明三,苗晋鑫.基于数据挖掘和网络药理学的中医药治疗动脉粥样硬化用药规律及特点分析[J].中药药理与临床.2021(10).
[7]王文鹃,相玉红,宋佳,张卓勇.基于他克林的乙酰胆碱酯酶抑制剂的 3D-QSAR 研究及虚拟筛选[J].化学研究与应用. 2014(02).
[8]知乎用户. 实现回归随机森林. https://zhuanlan.zhihu.com/p/52052903.2021(10)

附录

问题一	数据预处理	Python

```
# 读取数据文件
ADMET_train = pd.read_excel("./ADMET.xlsx",sheet_name="training")
ADMET_test = pd.read_excel("./ADMET.xlsx",sheet_name="test")
activity_train = pd.read_excel("./ERα_activity.xlsx",sheet_name="training",)
activity_test = pd.read_excel("./ERα_activity.xlsx",sheet_name="test")
molecular = pd.read_excel("./Molecular_Descriptor.xlsx")
molecular_test = pd.read_excel("./Molecular_Descriptor.xlsx",sheet_name="test")
desc_detail2 = desc_detail.fillna(method="ffill",axis=0)       # 填充
# 判断是否有缺失值
rawdata.columns[rawdata.isnull().any()].tolist() # 无缺失值列
rawdata.columns[rawdata.isna().any()].tolist()   # 无缺失值列
isfinite = pd.Series(np.isfinite(rawdata.drop(labels=["SMILES"],axis=1)).all())
isfinite[isfinite == False]        # 无无穷列
isinf = pd.Series(np.isinf(rawdata.drop(labels=["SMILES"],axis=1)).all())
isinf[isinf == True]
# 删除值一致的列
rawdata2 = rawdata.drop(rawdata.loc[:,rawdata.max(axis=0)==rawdata.min(axis=0)].columns,axis=1)
# 查找可能呈现泊松分布的变量(考虑到变量含义为 xxx 次数、个数。不编码)
freq_list = []
for col in rawdata2.columns:
      x = rawdata2.loc[:,col].value_counts().shape[0]
      if x <=10:
            print(col,x)
            freq_list.append(col)
import re
new_data_list = []
for data in freq_list:
      if    re.match('n.*', data) == None:
            new_data_list.append(data)
            print(rawdata2.loc[:,data].value_counts())
# 哑变量编码
xx = pd.get_dummies(rawdata2.loc[:,new_data_list])
# 数据标准化
zscore = preprocessing.StandardScaler()
data_scale = zscore.fit_transform(rawdata2.drop(labels=["SMILES","pIC50"],axis=1))
x_train = data_scale
y_train = rawdata2.loc[:,"pIC50"].values
features = rawdata2.drop(labels=["SMILES","pIC50"],axis=1).columns
train2 = pd.DataFrame(x_train,columns=features,index=rawdata2["SMILES"])
train2["pIC50"] = y_train
```

问题一	数据特征选择	Python

```
# 方差过滤
vt = VarianceThreshold()
vt.fit_transform(x_train)
var_thd = pd.DataFrame(vt.variances_, columns = ["Variance"], index=features)
var_thd = var_thd.reset_index()
var_thd.sort_values('Variance',ascending=0)

# F 检验过滤
f_model = SelectKBest(score_func=f_regression , k="all")
f_model.fit(x_train,y_train)
f_reg = pd.DataFrame(f_model.scores_, columns = ["FTest"], index=features)
f_reg = f_reg.reset_index()
f_reg.sort_values('FTest',ascending=0)

# 互信息过滤
test = SelectKBest(score_func=mutual_info_regression , k="all")
test.fit(x_train,y_train)
mutual_info = pd.DataFrame(test.scores_, columns = ["MutualInfo"], index=features)
mutual_info = mutual_info.reset_index()
mutual_info.sort_values('MutualInfo',ascending=0)

def GRA_ONE(DataFrame,m=0):
    gray= DataFrame
    std=gray.iloc[:,m]#为标准要素
    ce=gray.iloc[:,0:]#为比较要素
    n=ce.shape[0]
    m=ce.shape[1]#计算行列
    a=zeros([m,n])
    for i in range(m):
        for j in range(n):
            a[i,j]=abs(ce.iloc[j,i]-std[j])
    c=amax(a)
    d=amin(a)
    result=zeros([m,n])
    for i in range(m):
        for j in range(n):
            result[i,j]=(d+0.5*c)/(a[i,j]+0.5*c)
    result2=zeros(m)
    for i in range(m):
            result2[i]=mean(result[i,:])
    RT=pd.DataFrame(result2)
```

```
    return RT
gra = GRA_ONE(train2,m=train2.shape[1]-1)
gra_res = pd.DataFrame(gra.iloc[:-1,0].values, columns = ["GRA"], index=features)
gra_res = gra_res.reset_index()
gra_res.sort_values('GRA',ascending=0)

# 决策树
DT = DecisionTreeRegressor(random_state=1234)
DT.fit(x_train,y_train)
DT_inportance = pd.DataFrame(DT.feature_importances_, columns = ["DecisionTree"], index=features)
DT_inportance = DT_inportance.reset_index()
DT_inportance.sort_values(['DecisionTree'],ascending=0)

# XGBoost
xgboost = XGBRegressor(random_state=1234)
xgboost.fit(x_train,y_train)
xgboost_inportance = pd.DataFrame(xgboost.feature_importances_, columns = ["XGBoost"], index=features)
xgboost_inportance = xgboost_inportance.reset_index()
xgboost_inportance.sort_values(['XGBoost'],ascending=0)

# 随机森林变量重要性
forest = RandomForestRegressor(random_state=1234)
forest.fit(x_train, y_train)
RF_inportance = pd.DataFrame(forest.feature_importances_, columns = ["RandomForest"], index=features)
RF_inportance = RF_inportance.reset_index()
RF_inportance.sort_values(['RandomForest'],ascending=0)

# 极端随机树分类器
model = ExtraTreesRegressor(random_state=1234)
model.fit(x_train,y_train)
ET = pd.DataFrame(model.feature_importances_, columns = ["ExtraTrees"], index=features)
ET = ET.reset_index()
ET.sort_values(['ExtraTrees'],ascending=0)


# Lasso 正则化(系数取绝对值)
lasso = Lasso(alpha=0.00001,random_state=1234)
lasso.fit(x_train,y_train)
lasso_res = pd.DataFrame(np.abs(lasso.coef_), columns = ["Lasso"], index=features)
lasso_res = lasso_res.reset_index()
```

```
# 融合
all_res = [var_thd,f_reg,mutual_info,gra_res,DT_inportance,
                xgboost_inportance,RF_inportance,ET,lasso_res]
final_results = reduce(lambda left, right: pd.merge(left,
                                                    right,
                                                    on='index',
                                                    how='outer'),all_res)
final_results2 = final_results.loc[final_results["Variance"]>0,:]
final_results2.to_csv("./output/各方法的变量重要性粗筛.csv")
# 变量重要性可视化
fig=plt.figure(figsize=(40,25))
cols = final_results.columns.drop(["index","Variance"])
# plt.ylabel("Importance")

for i in range(len(cols)):

    ax=fig.add_subplot(1,len(cols),i+1)

    final_results3 = final_results2.sort_values(cols[i],ascending=False).head(50)
    coef_dt_sort0 = final_results3.sort_values(cols[i],ascending=True)
    index_sort =  final_results3[cols[i]].sort_values().index
    coef_dt_sort = final_results3.loc[index_sort,:]
    x, y = coef_dt_sort['index'], coef_dt_sort[cols[i]]
    map_vir = cm.get_cmap(name="Reds")
    color = map_vir(coef_dt_sort0[cols[i]])
    ax.barh(x, y, color=color,label=cols[i])
        ax.grid(linestyle="-.", axis='x', alpha=0.4)
    ax.set_xlabel(cols[i])
plt.subplots_adjust(wspace=0.1,hspace=0.05)
plt.savefig("./output/变量重要性可视化.png",dpi=500, bbox_inches = "tight")
plt.show()
# 最终变量选择结果
methods = final_results.columns.drop(["index","Variance"])
sort_var = dict()
for m in methods:
    sort_var[m] = final_results2.sort_values(m,ascending=False).head(50)["index"].values
  select_all = pd.DataFrame(sort_var)
select_all2 = select_all.melt()['value'].value_counts()
selec = select_all2.head(20)  # 综合选择的前 20 个变量
print(selec.index)
select_data_final = desc_detail2.loc[desc_detail2["Descriptor"].isin(selec.index.to_list()),:]
select_data_final.to_csv("./output/第一问选择的变量.csv ")
```

问题二	9 种预测方法交叉验证	Python

```
# 建立模型
X = train2.loc[:,selec]
print(X.shape)
y = train2.loc[:,"pIC50"]
# 划分数据集(可能需要按照时间顺序划分 Train、Test)
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=1234,test_size=0.2)
# 各个模型的交叉验证
models = {
      "Random Forest":RandomForestRegressor(random_state=1234, n_estimators=300),
      "XGBoost":XGBRegressor(),
      "AdaBoost":AdaBoostRegressor(),
      "GradientBoost":GradientBoostingRegressor(),
      "MLP":MLPRegressor(),
      "SVR_rbf":SVR(gamma='auto', kernel='rbf'),
      "Bagging":BaggingRegressor(),
      "Bag_KNN":BaggingRegressor(KNeighborsRegressor(), max_samples=0.5, max_features=0.5),
      "NuSVR":NuSVR(gamma='auto')

scores_dict = {}
for name,model in models.items():

      cv = ShuffleSplit(n_splits=10, test_size=.3, random_state=0)
      scores_CV = cross_val_score(model, X, y, cv=cv)
#         print(name,":",scores_CV)
      scores_dict[name] = scores_CV   # 保存交叉验证评分结果
df_scoresCV = pd.DataFrame(scores_dict)
df_scoresCV.to_csv("./output/交叉验证得分结果.csv",index=False,header=True)


# 可视化绘制箱线图
score_plotdata = pd.DataFrame(scores_dict).melt()

plt.figure(figsize=(15,8))
# sns.set(style="white")
color_p = sns.color_palette("Reds",9)
sns.boxplot(x="variable", y="value",data=score_plotdata)
plt.xticks(rotation=90)
plt.xlabel("Machine Learning Models")
plt.ylabel("Score")

plt.savefig("./output/交叉验证得分结果箱线图.png",dpi=500, bbox_inches = "tight")
plt.show()
```

问题二	RF,GBDT,XGBoost 的预测分析	Python

```
"""随机森林"""
rf = RandomForestRegressor(min_samples_leaf=2,
                           min_samples_split=4,
                           n_estimators=300,
                           random_state=1234)
rf.fit(X_train,y_train)
print(rf)
print('Training set score:{:.2f}'.format(rf.score(X_train,y_train)))
print('Test set score:{:.2f}'.format(rf.score(X_test,y_test)))
print("Training set MSE:{:.2f}".format(mean_squared_error(y_train,rf.predict(X_train))))
print("Test set MSE:{:.2f}".format(mean_squared_error(y_test,rf.predict(X_test))))

# 残差图
plt.plot((rf.predict(X_test)-y_test), linewidth=0.5)
plt.axhline(y=0, c="k", ls="--", lw=1)
plt.xticks([])
plt.xlabel("Test Sample")
plt.ylabel("RandomForest Residual")
plt.savefig("./output/RandomForest 模型拟合的残差图.png", dpi = 500,bbox_inches = "tight")
plt.show()

"""GBDT"""
gbr = GradientBoostingRegressor(learning_rate=0.1,
                                max_depth=4,max_features="log2",
                                n_estimators=200,
                               min_samples_split=2).fit(X_train,y_train)
print('Training set score:{:.2f}'.format(gbr.score(X_train,y_train)))
print('Test set score:{:.2f}'.format(gbr.score(X_test,y_test)))
print("Training set MSE:{:.2f}".format(mean_squared_error(y_train,gbr.predict(X_train))))
print("Test set MSE:{:.2f}".format(mean_squared_error(y_test,gbr.predict(X_test))))

# 残差图
plt.plot((gbr.predict(X_test)-y_test), linewidth=0.5)
plt.axhline(y=0, c="k", ls="--", lw=1)
plt.xticks([])
plt.xlabel("Test Sample")
plt.ylabel("GBDT Residual")
plt.savefig("./output/GBDT 模型拟合的残差图.png", dpi = 500, bbox_inches = "tight")
plt.show()
```

```
“”XGBoost””
xgb = XGBRegressor().fit(X_train,y_train)
print('Training set score:{:.2f}'.format(xgb.score(X_train,y_train)))
print('Test set score:{:.2f}'.format(xgb.score(X_test,y_test)))
print("Training set MSE:{:.2f}".format(mean_squared_error(y_train,xgb.predict(X_train))))
print("Test set MSE:{:.2f}".format(mean_squared_error(y_test,xgb.predict(X_test))))
# 残差图
plt.plot((xgb.predict(X_test)-y_test), linewidth=0.5)
plt.axhline(y=0, c="k", ls="--", lw=1)
plt.xticks([])
plt.xlabel("Test Sample")
plt.ylabel("XGBoost Residual")
plt.savefig("./output/XGBoost 模型拟合的残差图.png", dpi = 500, bbox_inches = "tight")
plt.show()
```

问题二	随机森林预测	Python

```
# 将随机森林应用到测试数据并输出结果
pIC50 = rf.predict(X_resdata)
IC50 = 1000000000 / 10 ** pIC50
answer2 = pd.DataFrame({"SMILES":X_resdata.index,"IC50_nM":IC50,"pIC50":pIC50})
answer2.to_csv("./output/问题 2 结果-随机森林预测.csv")
```

问题三	特征选择	Python

```
for ycol in ADMET_y:
    # 测试集数据标准化
    zscore = preprocessing.StandardScaler()
    ADMET_scale = zscore.fit_transform(concat_ADMET.drop(ADMET_y,axis=1))
    ADMETx_train = ADMET_scale
    ADMETy_train = concat_ADMET.loc[:,ycol].values
    features = concat_ADMET.drop(ADMET_y,axis=1).columns
    ADMETtrain = pd.DataFrame(ADMETx_train,columns=features,index=concat_ADMET.index)
    ADMETtrain[ycol] = ADMETy_train
     # 数据降维（根据随机森林计算的变量重要性筛选 50 个变量）
    forest = RandomForestClassifier(random_state=1234)
    forest.fit(ADMETtrain.drop(ycol,axis=1), ADMETtrain[ycol])
    RF_inportance = pd.DataFrame(forest.feature_importances_, columns = ["RandomForest"],
index=features)
    RF_inportance = RF_inportance.reset_index()
    RF_inportance.sort_values(['RandomForest'],ascending=0)
    sel = RF_inportance.head(50)["index"].values.tolist() # 筛选的变量名
    ycol_seldata[ycol] = sel
    y = ADMETtrain.loc[:,ycol]
    X = ADMETtrain.loc[:,sel]
```

问题三	模型交叉验证及混淆矩阵图	Python

```
    # 划分数据集
    X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=1234,test_size=0.8)
    # 多个模型交叉验证
    scores_dict = {}
    for name,model in models.items():
        cv = ShuffleSplit(n_splits=10, test_size=.3, random_state=0)
        scores_CV = cross_val_score(model, X,y, cv=cv)
        print(name,":",scores_CV)
        scores_dict[name] = scores_CV  # 保存交叉验证评分结果
    df_scoresCV = pd.DataFrame(scores_dict).round(3)
    df_scoresCV.to_csv("./output/交叉验证得分结果_{}.csv".format(ycol),index=True,header=True)

    # 交叉验证可视化绘制箱线图
    score_plotdata = pd.DataFrame(scores_dict).melt()
    plt.figure(figsize=(15,10))
    color_p = sns.color_palette(colormap2,9)
    sns.boxplot(x="variable", y="value",data=score_plotdata,palette=color_p)
    plt.xticks(rotation=90)
    plt.xlabel("Machine Learning Models")
    plt.ylabel("Score")
    plt.savefig("./output/交叉验证得分结果箱线图_{}.png".format(ycol),dpi=500, bbox_inches =
"tight")
    plt.show()
    # 建立各模型可视化结果
    fig, ax = plt.subplots(3,3,figsize=(20,15),sharey=True,sharex=True)
    flag = 0
    for name, model in models.items():
        clf = models[name].fit(X_train,y_train)
        train_predict = clf.predict(X_train)
        test_predict = clf.predict(X_test)

        # 利用热力图对于结果进行可视化
        confusion_matrix_result = metrics.confusion_matrix(test_predict,y_test)
        sns.heatmap(confusion_matrix_result,                                    annot=True,
cmap=colormap2,fmt='.50g',ax=ax[flag//3,flag%3])
        ax[flag//3,flag%3].set_xlabel(name)
        ax[flag//3,flag%3].set_ylabel("True Value")
        flag +=1
    # plt.subplots_adjust(wspace=0.1,hspace=0.1)
```

```
    plt.savefig("./output/所有模型混淆矩阵图_{}.png".format(ycol),dpi=500,bbox_inches = "tight")
    plt.show()
```

问题三	选择效果最优的模型对当前指标预测	Python

```
    # 模型应用到测试集
    model = df_scoresCV.mean(axis=0)[df_scoresCV.mean(axis=0)==df_scoresCV.mean(axis=0).max()].index[0]
    print("model:",models[model])  # 选择的优模型
    ycol_models[ycol] = models[model]

    # 测试结果数据标准化
    zscore = preprocessing.StandardScaler()
    ADMETtest_scale = zscore.fit_transform(concat_ADMET_test.drop(ADMET_y,axis=1))
    X_resdata = ADMETtest_scale
    testfeatures = concat_ADMET_test.drop(ADMET_y,axis=1).columns

    datatest = pd.DataFrame(X_resdata,columns=testfeatures,index=testrawdata["SMILES"])
    X_resdata = datatest.loc[:,sel]

    clf = models[model].fit(X_train,y_train)
    y_predict = clf.predict(X_resdata)
    ycol_pred[ycol] = y_predict
```

问题四	6 个随机森林预测模型构建	Python

```
class PredModel:
        # 随机森林
        if self.data_path == 'data/data_ER.xlsx':
                forest_reg = RandomForestRegressor()
        else:
                forest_reg = RandomForestClassifier()

        # 随机搜索优化
        param_grid = [{'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
                      {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]}]
        rand_search = RandomizedSearchCV(forest_reg, param_grid, n_iter=36, cv=5, scoring='neg_mean_squared_error')
        rand_search.fit(X, train_labels)
        final_model = rand_search.best_estimator_
        # 保存模型
        joblib.dump(final_model, self.model_path)

    def predFunc(input_from_data, mode):
        firstname = input_from_data.columns[0]
```

```
        X_input = input_from_data.drop(firstname, axis=1)
        y_input = input_from_data[firstname].copy()
        imputer = SimpleImputer(missing_values=0, strategy="median")
        scaler = StandardScaler()
        imputer.fit_transform(X_input)
        scaler.fit_transform(X_input)

        if mode == 0:
            SAVE_PATH = './model/forest_reg_ER.pkl'
        elif mode == 1:
            SAVE_PATH = './model/forest_reg_CAC.pkl'
        elif mode == 2:
            SAVE_PATH = './model/forest_reg_CYP.pkl'
        elif mode == 3:
            SAVE_PATH = './model/forest_reg_HER.pkl'
        elif mode == 4:
            SAVE_PATH = './model/forest_reg_HOB.pkl'
        elif mode == 5:
            SAVE_PATH = './model/forest_reg_MN.pkl'
        final_model = joblib.load(SAVE_PATH)

        # 进行预测
        predictions_input = final_model.predict(X_input)
        return predictions_input[0]
```

问题四	遗传算法主程序	Python

```
class GA:
    def __init__(self, path):
        self.df = pd.read_excel(path)
        self.result = []
        self.t = []
        self.record = []
        self.mutate_p = 0.05
        self.cross_p = 0.9

    def cul_single(self, person):
        other1 = person.columns[1]
        other2 = person.columns[2]
        other3 = person.columns[3]
        other4 = person.columns[4]
        other5 = person.columns[5]
        target_p = PredModel.predFunc(person.drop([other1,other2,other3,other4,other5], axis=1),
0)
```

```
true_target = person.iloc[0, 0]
print('本样本 ERa 值：', true_target)
print('预测模型认为的 ERa 的值', target_p)

# 初始化随机种群
buildings = []
value_record = []
s_record = []
best_param = [i*0 for i in range(5)] # 最佳参数位数
# best_param = [] # 最佳参数位数
percent_record = []
for i in range(60): # 为种群内个体数
    p = random.uniform(0.01, 0.99)
    temp = []
    for j in range(20): # 所有的特征属性数
        base_value = person.iloc[0, 6+j]
        # 变量调整范围为该变量的最大最小值
        right_value = define.limit[j][1] - person.iloc[0, 6+j]
        left_value = person.iloc[0, 6+j] - define.limit[j][0]
        if random.randint(0, 1):
            value = base_value + p * right_value
        else:
            value = base_value - p * left_value
        temp.append(value)
    # for j in range(19, 30): !
    buildings.append(temp)
for i in range(2):
    # 计算每个个体的适应度函数的值
    value, s_value,ADMET = Fit(buildings, person).fitness() # !
    temp_max = max(value)
    value_record.append(temp_max)
    index = value.index(temp_max)
    temp_s = s_value[index]
    temp_ADMET  = ADMET[index]
    s_record.append(temp_s)
    if temp_max > best_param[1]:
        best_param = [buildings[index],temp_max, temp_s, i,temp_ADMET]
    print('本轮最大 ERa 值：', temp_max)
    print('对应 ADMET 分数：', temp_s)
    print('================round', i, 'finished================\n')
    # Select 函数决定本代哪些个体可以活到下一代
    buildings_new = Select(buildings, value, s_value).selection()
    children = Cross(buildings_new, self.cross_p).crossover()
```

```
                # 子代以 0.3 的基准概率基因突变，突变概率动态降低
                buildings = Mutate(children, self.mutate_p).mutation()
            print('最佳变量：', best_param[0], '最优 Plc50 值：', best_param[1],
                  'ADMET 得分：', best_param[2], 'ADMET 各项指标:',best_param[4],'所在轮次:',
best_param[3])
            print(best_param)
            plt.title('Plc50 resault')
            plt.grid(ls='--')
            plt.xlabel("Iterations", fontsize=14)
            plt.ylabel("Plc50", fontsize=14)
            plt.plot(value_record)
            # plt.show()

            plt.title('ADMET score')
            plt.grid(ls='--')
            plt.xlabel("Iterrations", fontsize=14)
            plt.ylabel("ADMET score", fontsize=14)
            plt.plot(s_record)

            resultframe = pd.DataFrame([best_param])
            resultframe.to_csv('./result/opt_result_2021.csv', mode='a', index=False, header=None)
            # return best_param
        def run(self):
            # 提取最优解的基因型
            for idx in range(1):#self.df.shape[0]
            # for idx in range(132, 134):
                row = self.df[idx:idx+1][:]
                self.result.append(self.cul_single(row))
```

问题四	遗传算法中 pIC50 及 ADMET 预测	Python

```
# 适应度函数直接使用 5 个位置基因的加权总分
ER = 0
CAC = 1
CYP = 2
HER = 3
HOB = 4
MN = 5

class Fit:
    def __init__(self, buildings, person):
        self.buildings = buildings
        self.person = person
```

```
    def fitness(self):
        ER_value = []
        CAC_value = []
        CYP_value = []
        HER_value = []
        HOB_value = []
        MN_value = []
        tot_value = []
        ADMET = []
        otherER = self.person.columns[0]
        otherCAC = self.person.columns[1]
        otherCYP = self.person.columns[2]
        otherHER = self.person.columns[3]
        otherHOB = self.person.columns[4]
        otherMN = self.person.columns[5]
        # 设定不同位置基因的不同权重
        # 一号位基因即为 self.buildings[i][0]：1 代表 A11，2 代表 A12....5 代表 A15，其余以此类推
        for obj in self.buildings:
            for j in range(0, len(obj)):
                self.person.iloc[0, j+6] = obj[j]
ER_value.append(PredModel.predFunc(self.person.drop([otherCAC,otherCYP,otherHER,otherHOB,otherMN], axis=1), ER))

CAC_value.append(PredModel.predFunc(self.person.drop([otherER,otherCYP,otherHER,otherHOB,otherMN], axis=1), CAC))

CYP_value.append(PredModel.predFunc(self.person.drop([otherCAC,otherER,otherHER,otherHOB,otherMN], axis=1), CYP))

HER_value.append(PredModel.predFunc(self.person.drop([otherCAC,otherCYP,otherER,otherHOB,otherMN], axis=1), HER))

HOB_value.append(PredModel.predFunc(self.person.drop([otherCAC,otherCYP,otherHER,otherER,otherMN], axis=1), HOB))

MN_value.append(PredModel.predFunc(self.person.drop([otherCAC,otherCYP,otherHER,otherHOB,otherER], axis=1), MN))

        for i,j,m,n,p in zip(CAC_value,CYP_value,HER_value,HOB_value,MN_value):
            ADMET.append([i,j,m,n,p])
        for i,j,m,n,p in zip(CAC_value,CYP_value,HER_value,HOB_value,MN_value):
            if m == 0: # HER 和 MN 取 0 时反而好，为计算方便将其转换
                m = 1
```

```
            elif m ==1:
                m = 0
            if p == 0:
                p = 1
            elif p == 1:
                p = 0
            summ = i+j+m+n+p
            tot_value.append(summ)
        for i,j in enumerate(tot_value):
            if j < 3:
                ER_value[i] = -1000000
        # 返回种群中每个个体的基因型权值和，即适应度
        return ER_value, tot_value, ADMET
```