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	新疆大学
参赛队号	21107550064
队员姓名	1.王海
	2.范鑫
	3.曹青青

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 乳腺癌 ERα拮抗剂的优化建模

摘 要：

乳腺癌是当下最常见、致死率较高的癌症之一，ERα被认为是治疗乳腺癌的重要靶标，将能够拮抗 ERα活性的化合物作为治疗乳腺癌的候选药物。但在药物研发中，一个化合物若要成为候选药物，不仅需要具备良好的生物活性，还需要具备良好的 ADMET 性质。因此，如何利用所提供的 ERα拮抗剂信息，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，为同时优化 ERα拮抗剂的生物活性和 ADMET 性质提供预测服务，是研究化合物为治疗乳腺癌候选药物的关键。

针对问题一，本文建立了基于随机森林的重要性选择模型，通过该模型进行了分子描述符的初步筛选，选择出了前 40 个对生物活性重要的变量。再通过皮尔逊相关系数分析 40 个变量间的耦合性，对皮尔森相关系数大于 0.6 的变量组，按照重要性排序进行剔除，最终留下了 20 个对化合物生物活性最具显著影响且彼此相对独立的分子描述符。

针对问题二，本文利用问题一已经筛选出来的 20 个主要分子描述符，采用五种机器学习模型作为化合物对 ERα生物活性的定量预测的初步模型，然后选取预测性能最好的模型对文件“ERα_activity.xlsx”中 test 表的化合物进行 IC_{50} 值和 pIC_{50} 值预测，通过多种性能指标（MSE、R^2 等）比较，最终选取随机森林来作为生物活性最终预测模型。

针对问题三，对于题目所提的分子描述符和 ADMET 性质数据，首先判定该问题是建立二分类预测模型，经分析，本文提供了三种解决方案：方案 1，针对五个 ADMET 建立五个分类器；方案 2，建立多标签分类模型；方案 3，建立多任务学习分类预测模型。通过实验分析，前两种方案中方案 1 分类预测效果更好，因方案 3 存在众多不确定性，故本文选择方案 1 作为问题三的最终解决方法，方案 3 作为模型改进与推广。针对方案 1，本文选取五种机器学习经典分类模型算法分别对 ADMET 性质进行分类预测，通过分类模型性能指标对比，最终选取梯度提升树对以下三种 ADMET 性质进行分类预测：Caco-2，hERG，HOB；选取随机森林以下两种 ADMET 性质进行分类预测：CYP3A4，MN。

针对问题四，因数据输入维度的原因，本文在问题四用表现优异的随机森林算法重新建立了 5 个 20 维变量的 ADMET 分类器。通过粒子群智能优化搜索算法进行 20 个变量的最优取值搜索，并且创新提出通过将分类器判别结果嵌入粒子群适应度函数的方法大大降低了整体搜索的时间成本，通过 50 轮迭代即取得了最优值。

本文最后对所建立模型进行分析、评价和改进。本文**最大的优点**在于以下几部分：

(1) 本文将建模与化合物的现实相结合，关注并分析了实际药物研发过程中典型的分子描述符；

(2) 本文在建模过程中，将机器学习中的经典回归、分类模型灵活应用，并产生了很好的

性能效果；

(3) 本文利用粒子群搜索算法智能的解决优化问题，搜索速度快、参数少、效率高，规避了人工求解优化问题的缓慢性、机械性和局限性；

(4) 本文分析了在现实应用场景中应多关注的 ADMET 分类模型性能评价指标：精确度（Precision）、召回率（Recall）。

关键词：癌症靶点 随机森林 梯度提升树 皮尔逊相关系数 粒子群智能搜索算法

目录

1 问题重述

1.1 问题背景

目前，乳腺癌是世界上得病率及致死率都较高的癌症之一，是一激素依赖性肿瘤，其的发生发展与激素受体的表达密切相关，其中与雌激素受体的表达关系最为密切[1]。有研究表明，用免疫组化法在正常乳腺上皮中可检测到少量 ERα，几乎所有导管上皮不典型增生病灶都高表达 ERα，大约 75%的导管原位癌、90%以上的小叶原位癌、70%的浸润性乳腺癌病灶中可检测到明显升高 ERα，并且表达不存在年龄差异[2］，这些结果提示 ERα在乳腺癌的发生发展中可能起到促进作用。因此，ERα被认为是治疗乳腺癌的重要靶标，将能够拮抗 ERα活性的化合物作为治疗乳腺癌的候选药物。

随着生信和计算机技术的发展，对化合物结构与其活性之间关系的定量描述研究，即 QSAR（Quantitative Structure-Activity Relationship）研究，在各个领域中的应用日渐成熟。在药物的研发过程中，往往会利用收集到的一系列与疾病相关靶标的化合物及其生物活性数据，建立 QSAR 化合物活性预测模型来预测筛选具有更好活性的生物化合物分子，这对于药物分子设计和先导化合物改造具有极其重要的意义。定量构效关系(QSAR)的诞生，目的是要替代传统耗时且昂贵的高通量筛选（HTS），其通过参考活性与非活性化合物的结构信息来助力推出更优的化合物结构。

图 1.1　QSAR 概念图[9]

当然，具有良好活性的化合物分子作为候选药物是远远不够的，还必须从人体本身对化合物的反应出发，对化合物成药性和安全性进行评估，即对 ADMET 性质进行预测。ADMET 是衡量化合物成药性的重要指标参数，主要包括药物的吸收（Absorption）、分布（Distribution）、代谢（Metabolism）、排泄（Excretion）和毒性（Toxicity），其中，药物的吸收、分布、代谢、排泄合称为化合物药代动力学性质，简称 ADME，其主要阐述了化合物在生物体中的浓度随着时间变化的规律；毒性（Toxicity）是指药物对于生物体的毒害程度。上世纪 90 年代之前，化合物的 ADMET 性质不佳导致药物研发失败的占比达到

39%[3]，故在药物开发早期，甚至在化合物发现阶段就应该对化合物的 ADMET 性质进行评价，从而选择 ADMET 性质较好的化合物作为候选药物，可在一定程度上缓解药物研发实验的经济压力，并有效提高候选药物后期开发的成功率。

综上所述，如何利用治疗乳腺癌的重要靶标 ERα的拮抗剂信息数据，构建 QSAR 化合物活性预测模型和化合物 ADMET 性质的分类预测模型，对治疗乳腺癌药物研发早期的选化合物作为候选药物极其关键，性能优良的分类预测模型能够对候选化合物进行有效的药性评价和风险评估，从而提高治疗乳腺癌的药物研发成功率，节省时间和经费的投入。

1.2 问题重述

基于上述研究背景，本文需研究和解决以下问题：

问题一：分子描述符筛选

参考文件“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），并请详细说明分子描述符筛选过程及其合理性。

问题二：化合物活性预测模型

结合问题 1 所选择不超过 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，请叙述建模过程。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 IC50 值和对应的 pIC50 值预测，并将结果分别填入“ERα_activity.xlsx”的 test 表中的 IC50_nM 列及对应的 pIC50 列。

问题三：化合物 ADMET 性质分类预测模型

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。然后使用所构建的 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：分子描述符的取值优化

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 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 模型假设

假设 1：乳腺癌治疗靶点的选择可靠，针对其设计的药物能真实影响靶点生物活性；
假设 2：所有模型的构建中，训练集和测试集属于同一分布；
假设 3：本文主要考虑分子描述符所代表的低维信息，不考虑其代表的三维等高维信息；
假设 4：同一题所有预测模型的训练均采用相同维数及维度，默认所选特征为最佳特征；
假设 5：所有数据均为原始数据，来源可靠，采集科学；
假设 6：所选特征值经过标准化处理，不会影响模型效果。

3 符号说明

序号	符号	含义
1	MSE	均方误差
2	R^2	判定系数
3	Acc	精确度
4	Pre	预测率
5	Rec	召回率
6	F1_score	F1 分数
7	AUC	分类中 ROC 曲线下的面积
8	P	种群大小
9	D	维度
10	ω	权重因子
11	c	学习因子
12	t	最大迭代次数
13	x	初始化粒子的位置
14	v	初始化各种粒子的速度

4 化合物特征选取

4.1 问题一分析

本问题要求根据文件“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）。

分子描述符是用于研究结构-性质相关性方法学的重要部分，在分子设计中化合物优化应用。描述符的合理选择对于基于 QSAR 模型的药物开发非常重要，可在优化药效学和药代动力学特性的同时，减少不必要的实验数量。

为从 1974 组样本变量中选取建模的主要变量，需要对数据进行降维处理，然后对降维后的数据变量进行相关性分析，对相关性较高的变量，选择部分进行剔除，继而得到 20 个相互独立的、具有显著性影响的主要变量。为较好地筛选主要变量，首先，采用机器学习算法中适用于处理非线性关系的随机森林对样本数据进行分析，得出了分子描述符（即变量）对生物活性影响的重要性排序。初步选取排序前 40 的变量。其次，考虑到某些变量之间高度相关，为保证选取的变量具有独立性，采用距离相关系数对 40 个变量进行分析，逐步剔除 40 个变量之间相关性较高的变量，最终选取了 23 个主要变量，另外从工程应用角度来看，降维后的变量个数应小于等于 20，故选取前 20 个对生物活性最具有显著影响的变量最为主要变量。最后，对选取的 20 变量计算两两相关性，结果表明，选取的变量之间关系较弱，具有很好的独立性。

本题的难点在于：

（1）各参数变量与抑制 Erα生物活性值之间具有高度非线性关系，判定因、自变量相关程度较为困难，同时为了后续的操作条件优化，选择的变量必须是原有变量，这是特征选择问题，无法使用较为常规的特征提取方法；

（2）由于变量过多，且变量与变量之间可能存在相互强耦连的关系，故选取相互独立的变量较难处理。

针对难点 1，主要变量代表性问题，数据降维算法分为特征选择与特征变换两类。特征选择为从给定的特征中直接选择若干重要特征，特征变换为通过某种变换将原始输入空间数据映射到一个新空间中。采用基于有监督机器学习的随机森林算法获取到各变量对活性值贡献度的排名，依此实现对选取变量代表性的判断。

针对难点 2，高耦合变量独立性问题，根据随机森林得到变量的贡献度排名后，依据从高到低的顺序对变量进行基于距离相关性系数的高相关性滤波，过滤掉耦合变量，以此实现提取具有独立性的特征变量。

最后，本文只要从上述的两个方面出发，筛选出前 20 个对生物活性最具有显著影响的分子描述符（即变量）。问题一的思路流程图如图 4.1 所示：

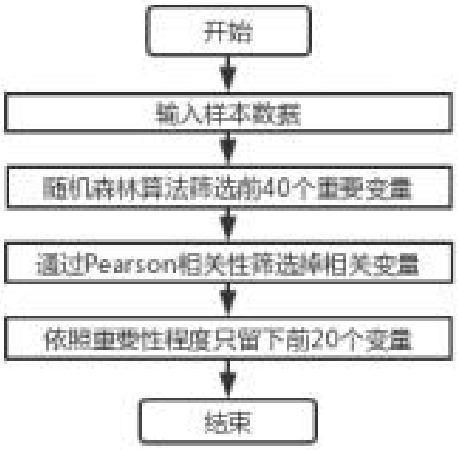


图 4.1 问题一思路流程图

4.2 变量降维

4.2.1 随机森林算法——评价变量重要度

随机森林（Random Forests，简称 RF）[4]是 Bagging 的一个扩展变体。RF 在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。

1、基本原理

随机森林通过 bootstrap 重采样技术，从原始数据集中反复随机选取 s 个样本，生成一个新训练样本集来训练决策树，然后重复上述步骤 t 次，最终便可以生成 t 个决策树，形成一个随机森林，新数据的分类结果则是按照由分类树投票多少所形成的分数而定

特征选取使用随机方法，然后对比在不同情况下所产生的误差。虽说单棵分类树的分类效果可能很小，但在通过随机的方法产生大量的决策树后，一个测试样品会经过森林中每棵树的分类结果，经对所有树统计计算后选择可能性最大的类别。

2、构造过程

（1）从原始数据的训练集中通过 Bootstraping 方法随机有放回的进行采样，最终选出 t 个样本，共进行 s_tree 次采样，生成 s_tree 个训练集；

（2）对 s_tree 个训练集，分别训练 s_tree 棵决策树模型；

（3）对于单棵决策树模型，假设训练样本特征数为 s，那么每次分裂时根据信息增益/信息增益比/基尼指数选择最好的特征进行分裂；

（4）每棵决策树都要一直这样分裂下去，直到该节点的所有训练样本数据都归为同一类别。在决策树分裂过程中不需要剪枝；

（5）将所产生的多棵决策树组合成一个森林，即随机森林。对于分类问题：按多棵树分类器投票决定最终分类结果；对于回归问题：由多棵树预测值的均值决定最终预测结果。

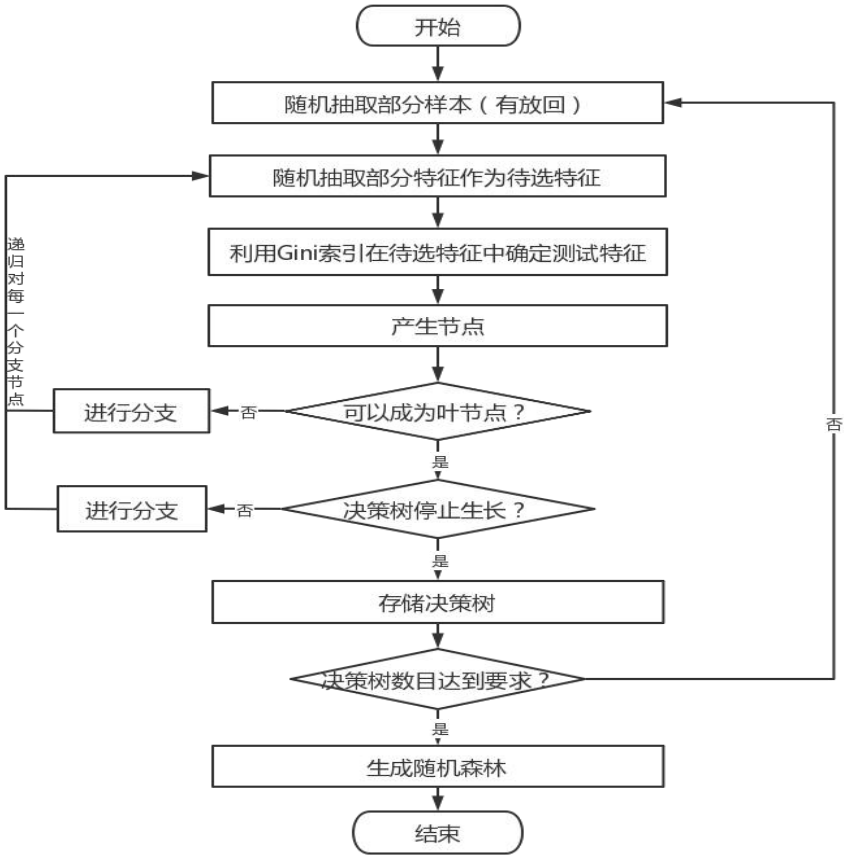


图 4.2 随机森林流程图

本题通过随机森林算法将 729 个分子描述符按照与抑制靶点活性值的关联重要程度进行排序，获得前 40 个最重要的分子描述符，如表 4.1 所示：

表 4.1　40 个变量及其重要程度指数

序号	分子描述符	重要性指数
1	MDEC-23	0.1948
2	LipoaffinityIndex	0.0531
3	maxHsOH	0.0398
⋮	⋮	⋮
40	MLFER_S	0.0038

4.2.2 过滤高相关性变量

由于本问题所提供的数据变量具有非线性和高耦合性的特征，故需对随机森林提取出

的前 40 个变量进行相关性分析，过滤掉一些相关性较高的变量。在互相高度相关的变量中，本文保留贡献值最大的变量，认为它能够代表所在高耦合变量组的全部信息。首先，对随机森林提取出来的前 40 个变量做了变量间相关性热图，如下图 4.3 所示：

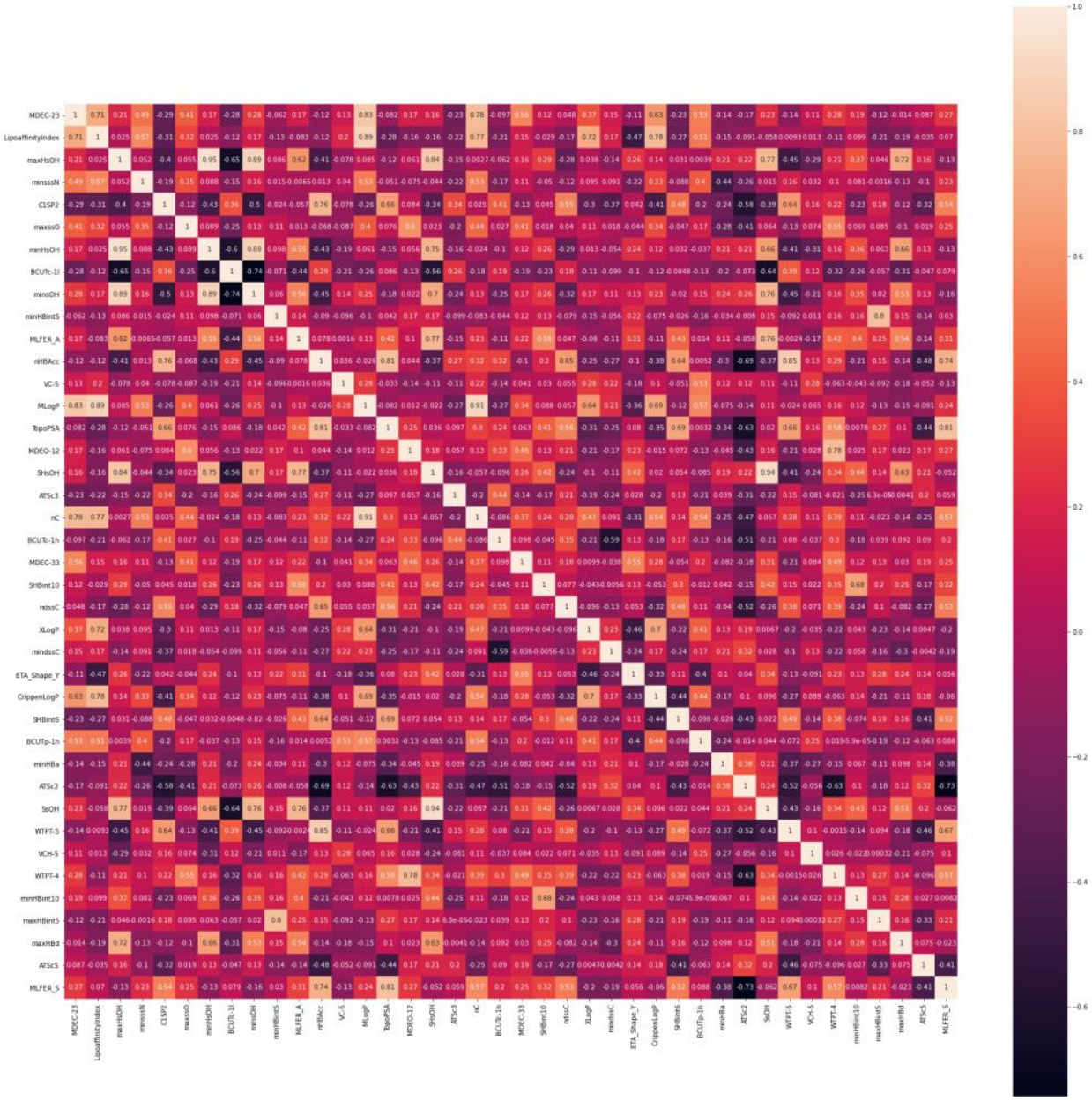


图 4.3 前 40 变量之间相关性热力图

从上图 4.3 可明显看出，通过随机森林获得的前 40 个变量之间存在一定的耦合性，因此对这 40 个变量中相关性较高的变量有必要进行剔除。

在统计学中，Pearson 乘积矩相关系数（Pearson product-moment correlation coefficient，又称作 PPMCC 或 PCCs，常用 r 表示）用于度量两个变量 X 和 Y 之间的相关（线性相关），其值介于-1 与 1 之间。在自然科学领域中，该系数广泛用于度量两个变量之间的相关程度。它是由卡尔·皮尔逊从弗朗西斯·高尔顿在 19 世纪 80 年代提出的一个相似却又稍有不同的想法演变而来的[8]，其计算过程如下。

两个变量之间的皮尔逊相关系数定义为两个变量之间的协方差和标准差的商，

$$\rho_{X,Y}=\frac{\operatorname{cov}(X,Y)}{\sigma_X\sigma_Y}=\frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}. \tag{4.1}$$

以上方程定义了总体相关系数, 一般表示成希腊字母 $\rho(rho)$。基于样本对协方差和标准差进行估计，可以得到样本相关系数, 一般表示成 r：

$$r=\frac{\sum_{i=1}^{n}(X_i-\overline{X})(Y_i-\overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\overline{Y})^2}}. \tag{4.2}$$

其中 $\overline{X}$， σ_X 分别是样本平均值和样本标准差。

皮尔森相关系数的值介于-1 与+1 之间，即－1≤r≤+1，其性质如下：

表 4.2 皮尔森相关系数性质

条件	含义
r<0	两变量正相关
r>0	两变量为负相关
|r|=1	两变量为完全线性相关，即为函数关系
r=0	两变量间无线性相关关系
0<|r|<1	两变量存在一定程度的线性相关；且|r|越接近 1，两变量间线性关系越密切；|r|越接近于 0，表示两变量的线性相关越弱。

其中，皮尔森相关系数一般可按三级划分：|r|<0.4 为低度线性相关；0.4≤|r|<0.7 为显著性相关；0.7≤|r|<1 为高度线性相关。

根据题目要求，去耦合后的变量之间应具有中相关及以下的相关性，故将距离相关系数的阈值设为 0.6。基于此，对表中的变量进行相关性检验。首先计算两两变量间的距离相关系数，然后判断两者的相关系数是否大于所设阈值。若大于，则说明两变量相关性高，对两变量间贡献值排名靠后的那一变量采取删除操作。若小于，则说明两变量间并不强相关，则将两变量都暂时予以保留。之后重复上述操作，直至遍历结束，最终筛选出来的 20 个变量（详见附录），其热力图如下图 4.4 所示，可以明显看到经过相关性筛选后的分子描述符变量的独立性大大增强。

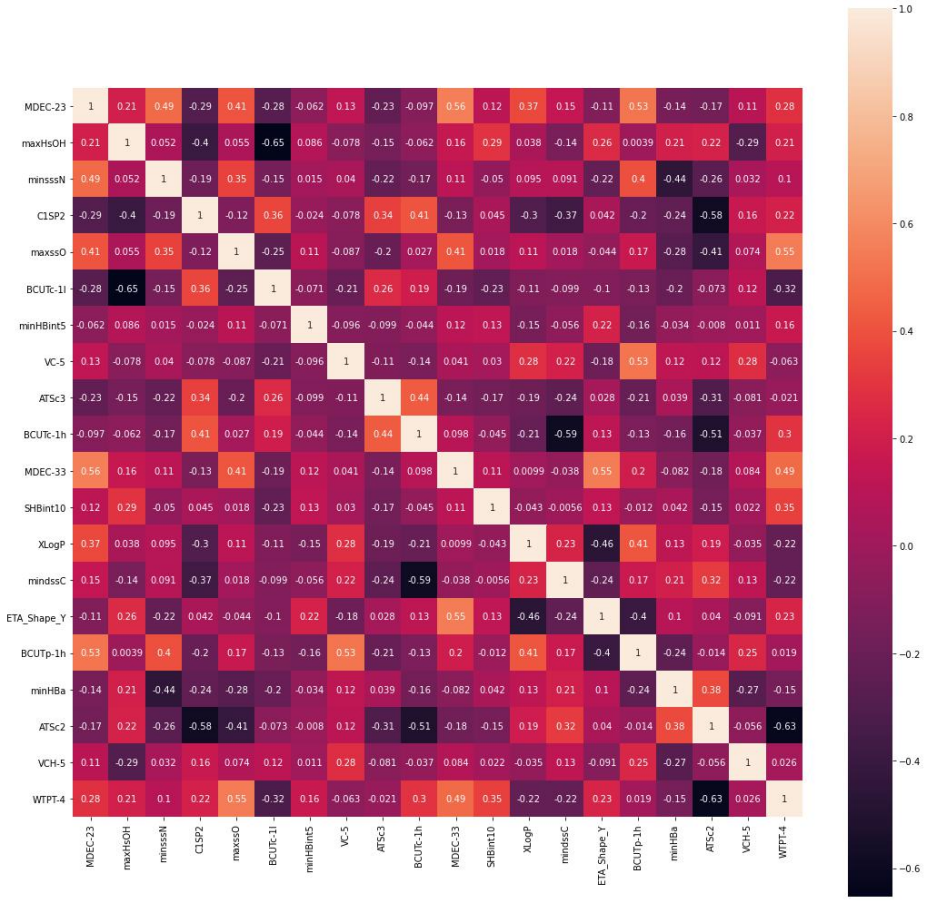


图 4.4 最终筛选的 20 个分子描述符及热力图

4.3 变量选择合理性评价

从变量降维过程中采用的算法及处理流程来看，根据随机森林得出的变量贡献度排名，保留排名靠前的变量从而保证了所选取变量与因变量之间的相关性，而设计的基于距离相关性高相关度变量滤波算法则保证了降维后变量之间的独立性，可根据所提取的特征变量之间的距离相关系数计算结果对变量之间的相关程度进行可视化，如图 4.4 所示。可见选取的 20 个变量之间相关性低，独立性较好。

从变量降维后的实际背景来看，该问题的关键是筛选出对生物活性最具影响的分子描述符，在通过随机森林算法实现对于描述符重要性的排序后，筛选出了最为重要的 40 个分子变量，再通过相关性分析找到这些变量之间的相关程度，通过将相关性较大的一组变量中留下一个来，从而筛选掉一些变量。最终留下的 20 个变量分别属于不同的分子特性类别，在实际药物的研发中也极大程度上代表了分子的各方面性质，保证了药物研发的科学性。

5 化合物活性预测模型

5.1 问题二分析

问题二要求采用问题一中所筛选的不超过 20 个对生物活性最具有显著影响的分子描述符（即变量），建立化合物对 ERα生物活性的定量预测模型，同时，利用所建立的预测模型对“ERα_activity.xlsx”文件中 test 表的化合物进行 IC_{50} 值和 pIC_{50} 值预测，并将结果分别填入所对应的位置。该问题的难点在于如何利用所筛选出的 20 个分子描述符数据建立效果良好的预测模型。

随着大数据时代的到来，机器学习方法的应用越加广泛，也更加成熟。机器学习算法机器学习中定义机器学习（Machine Learning）本质上就是让计算机自己在数据中学习规律，并根据所得到的规律对未来数据进行预测[5]。从数据的结构出发，化合物的构效关系一般具有非线性的、高维度的等特性，传统 QSAR 模型对化合物活性预测进行建模具有一定挑战。因此，本文利用多种机器学习模型来建立对化合物活性预测的 QSAR 模型。首先，将样本数据集数据进行标准化处理；其次，采用从 1974 组样本数据中抽取 70%的数据用于多种机器学习模型训练，使用剩下的 30%的数据对模型进行验证和评价；最后，选取多种机器学习模型中性能最好的模型，来对“ERα_activity.xlsx”文件中 test 表的化合物进行 IC_{50} 值和 pIC_{50} 值预测。问题二的思路流程图如图 5.1 所示：

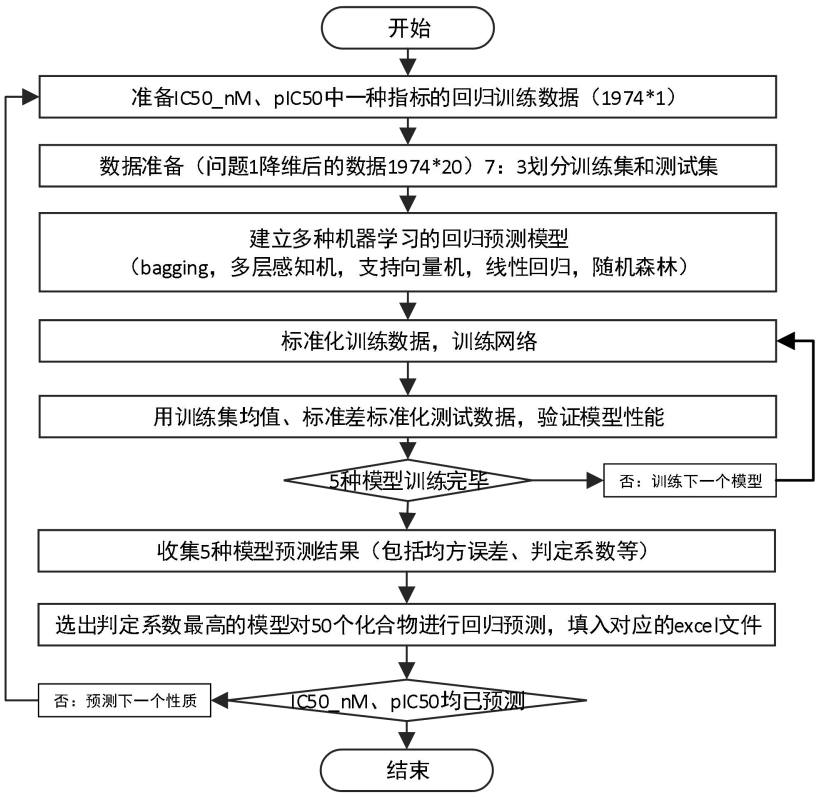


图 5.1 问题二思路流程图

5.2 化合物活性预测模型的建立

5.2.1 多种机器学习回归预测模型基本原理

本问题主要基于机器学习方法对 ERα拮抗剂的化合物生物活性构建定量预测模型，其中，使用到多种机器学习方法，分别是：多层感知机、线性回归、Bagging 算法、支持向量机、随机森林。

一．多层感知机

MLP(Multi-layer Perceptron)多层感知机是一种具有前向结构的人工神经网络。多层感知机可以解决非线性和可分离的问题。

1、MLP 概念

MLP 多层感知机是一种含有前向结构的 ANN，其主要是将一组输入向量映射到一组输出向量，它可以被视为由节点层组成的有向图，且每一层都完全连接到下一层。其中，除输入节点以外，该网络中的其它节点都是具有非线性激活函数的神经元，在训练 MLP 时，一般采用监督学习中的反向传播算法。多层感知器是感知机的一种推广，它将感知机中无法识别不能线性分离的数据的弱点有所攻克。

若将其与单层感知器相比，MLP 的输出端从单个变成了多个；不再是输入和输出两端之间只有一层，而且还多增加了两层：一个输出层、一个隐藏层。该网络结构中的信息处理方向是逐层的，从输入层到隐藏层再到输出层。MLP 是一种典型的，基于反向传播学习的前馈网络，其使用交叉熵损失函数所生成的误差来训练网络中各层的参数从而来反映数据的实际情况。隐藏层主要将输入空间中的数据进行非线性映射，输出层则主要用来提供线性分类，通过这两层，我们可以同时训练学习非线性映射和线性判别函数。图 5.2 显示了使用一个隐藏层的多层感知机神经网络结构图：

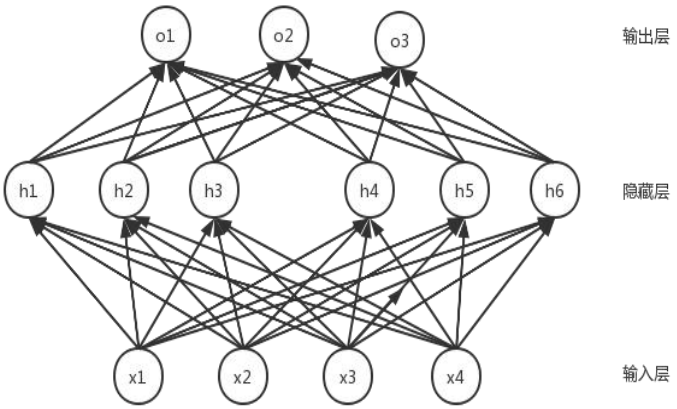


图 5.2 含有一个隐藏层的多层感知机网络结构图

其中，输入层、隐藏层和输出层分别有 4、6 和 3 个神经元，由于输入层只是输入数据，不参与网络的计算，所以图 5.2 中所展示的 MLP 层数是 2 层。由图 5.2 可以看出，隐藏层中每个神经元都与输入层的输入全连接，输出层中每个神经元都与隐藏层中的全连接，因此 MLP 各层网络之间都是用全连接层相互连接。

2、MLP 训练过程大致如下：

（1）随机分配每个边权值；

（2）前向传播：输入层使用训练集中所有样本数据特征，激活训练集中全部输入 ANN，通过前向传播获得输出值；
（3）反向传播：计算所有输出值和真实值的误差并总和，然后使用反向传播算法来对权重进行更新；
（4）对 2)~3)步骤重复，直到输出结果满足人为设定的限制条件。
通过对上述过程的实现，就可以获得一个训练好的 MLP 网络模型，此时，该网络可用来进行其他相关问题的试验。

二．线性回归

线性回归是数理统计中广泛使用的一种统计学方法，它使用回归分析来确定两个或多个变量之间的定量关系。 线性回归可以分为两种类型，具体分类是取决于所考虑的变量数量是否唯一： 一元线性回归和多元回归。本文使用多元线性回归建立化合物活性预测模型，将问题一中所筛选出的不超过 20 个分子描述符作为自变量，文件“ERα_activity.xlsx”的 train 表中的化合物 IC_{50} 值和对应的 pIC_{50} 值为两个因变量，对两个因变量分别建立多元线性回归模型。

在进行多元线性回归模型的建立时，首先要注意自变量的选择，以确保所建立的回归模型拥有有较好的解释和预测效果。 一般选取准测如下：
（1）所研究自变量必须对因变量含有特定的显著影响；
（2）每个变量应该是互斥的，即自变量之间的关联度不能高于自变量与因变量之间的相关程度；
（3）自变量数据必须统计完整。

三．Bagging 算法

Bagging 算法（Bootstrap aggregating，引导聚集算法），也称为装袋算法，是机器学习领域的一种集成学习算法。最初由 Leo Breiman 于 1996 年提出。Bagging 算法可以通过与其他分类、回归算法贯穿，从而来提升模型的准确程度、稳定程度，同时，也可通过将结果方差降低，来规避过拟合现象的发生，在结果方差降低方面，其主要是通过组合多个模型的形式来做，这种思维技术方式一般称为集成学习。

四．支持向量回归

支持向量回归（SVR）是所属支持向量机（SVM）下的一个分支，二者的主要差别在于：对于 SVM，模型训练希望所学习到的超平面，其离最近样本点的“远近”程度最大；而对于 SVR 则是希望该平面离其最远样本点的“远近”程度最小。SVR 的主要原理图及释义如下图 5.3 所示：

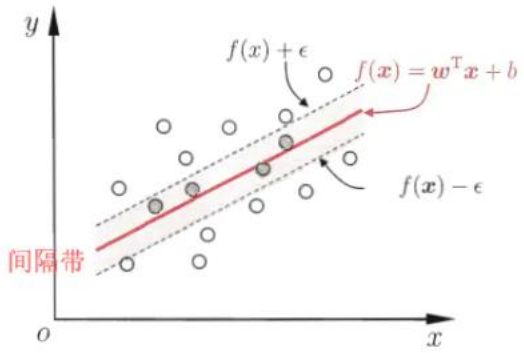


图 5.3 支持向量回归示意图

据上图可以发现，SVR 主要是在一个线性函数的两边创建“间隔带”，其中的间隔为 ε(也叫容忍偏差，是一个需要人为定义的经验值)，当样本训练时若落在所设定的间隔区域

内部，则模型不计算损失值，也就是说会对 SVR 函数模型训练生成相应影响的只有支持向量，然后计算所有支持向量的损失和，通过以下两个优化目标来获得训练好的 SVR：总损失最小化、间隔最大化。

五．随机森林

随机森林在一定程度上与上述 bagging 算法类似，它会对原始数据执行反复多次采集样本，每次采集和样本量相同的观测数据，由于进行的是有放回采样，故可能会存在以下状况：有部分的观测值每次都采集不中，而有些观测值则会重复提取，经过上述操作则会得到多个不一样的的数据集，然后为各个数据集建立一棵决策树，继而会生成大量的决策树来形成森林。

值得更多注意的是，林中的每个决策树相互之间是不具有相关性的，其最终的输出结果由林中的每个决策树决定。在利用随机森林处理回归问题时，它是将林中每个决策树结果的平均输出作为最后的输出结果。随机森林的基本单元是决策树，它通过选择信息熵增益最大的特征来构造决策树，以适应高维数据的混沌分布。本文在利用随机森林建模时，也主要是借助这一特性来更好的做回归预测。

5.2.2 化合物活性预测模型建立

在化合物活性预测模型的建立中，本文主要采用上述所介绍的五种机器学习方法：多层感知机、线性回归、Bagging 算法、支持向量机、随机森林，本问题首先利用文件“Molecular_Descriptor.xlsx”和文件“ERα_activity.xlsx”中 train 表中数据对这五种机器学习方法模型进行训练和测试；其次，将五种机器学习模型的测试结果进行比较评估；最后，选取五种机器学习模型方法中预测性能最好的模型来做本问题最终建立的化合物活性预测模型，继而利用该模型对文件“ERα_activity.xlsx”中 test 表中化合物活性进行预测并将预测值填入表中。

一．数据准备

此步中需对文件“Molecular_Descriptor.xlsx”中的 1974 组数据和文件“ERα_activity.xlsx”中的化合物活性数据进行整理，保留问题一所提取出的 20 个对生物活性最具有显著影响的分子描述符及化合物活性表示 IC_{50}、pIC_{50} 值，将整理后的数据集按照 70%和 30%分为模型的训练集（70%）和测试集（30%），然后用 python 对训练和测试模型所用到的数据进行以下标准化处理：

对序列 $x_1, x_2, \cdots, x_n$ 进行变换：

$$y_i = \frac{x_i - \bar{x}}{s}. \tag{5.1}$$

其中，

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}\ . \tag{5.2}$$

则新序列 $y_1, y_2, \cdots, y_n$，即标准化处理后的数据均值为 0，方差为 1，且无量纲。

同时，为了满足训练集和测试集数据属于同一数据分布这一假设，在对训练集和测试集数据进行标准化时，用的都是由训练数据集生成的均值 $\bar{x}$ 和标准差 s。

二．化合物活性预测模型建立

利用整理好的数据对上述所提到的五种机器学习模型进行训练测试，根据回归指标选

出模型测试效果性能最优的，整体流程框架如图 5.4 所示：

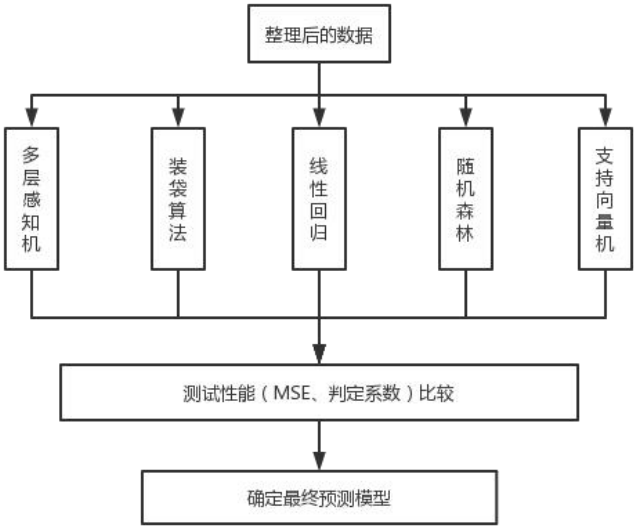


图 5.4 最优模型确立流程框架图

5.3 模型求解及结果分析

本文采用机器学习库 sk-learn 对上述模型进行实现，并采用均方误差、判定系数作为模型性能评价指标，详情如下：

1、均方误差（Mean Squared Error，MSE）：

该指标旨在计算预测值和真实值对应样本点之间误差的平方和均值，该值越大则说明拟合效果越差，计算公式如下：

$$MSE = \frac{1}{n}\sum_{i=1}^{n}[f(x_i) - y_i]^2 . \qquad (5.3)$$

2、判定系数（R^2 score，R^2）：

该指标旨在计算所建立的回归模型解释方差得分，其值取值范围在区间[0,1]内，若越接近于 1 则说明自变量越能够去解释因变量的方差变化，值越小则说明效果越差，计算公式如下：

$$R^2 = 1 - \frac{MSE(y_i, f(x_i))}{\frac{1}{m}\sum_{i=1}^{m}(y_i - \bar{y})^2} . \qquad (5.4)$$

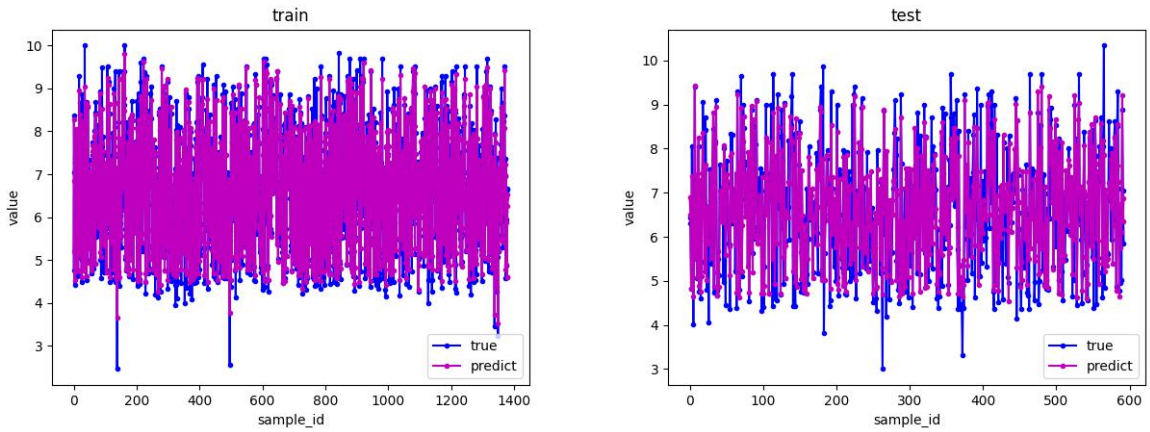
经过试验，将各个模型所获得的性能指标值进行汇总，模型全部性能预测指标结果见附录，此处只展示两种评价指标结果，详情如表 5.1。

表 5.1 五种机器学习模型的 MSE、R^2 性能评估

	MSE		R^2	
	Train_data	Test_data	Train_data	Test_data
随机森林	**0.0818**	**0.5297**	**0.9596**	**0.7385**
支持向量机	0.0921	0.7122	0.9545	0.6485
Bagging 算法	0.1095	0.6494	0.9459	0.6794
多层感知机	0.1573	0.7237	0.9223	0.6428
线性回归	0.8767	0.9212	0.5667	0.5453

由上表 5.1 可以看出，在评价指标 MSE 下，不管对于训练集还是测试集，随机森林的 MSE 都是五种机器学习算法模型中最低，性能最优的；在评价指标 R^2 下，不管对于训练集还是测试集，随机森林的 R^2 都是五种机器学习算法模型中最高，性能最优的；综上所述，不管是评价指标 MSE 还是评价指标 R^2，随机森林的测试效果整体来看都是不错的，尤其在测试数据集中。

另一方面，为直观看出随机森林预测模型效果好坏，将预测得到的结果与其真实值进行对比，结果如图 5.5 所示。



(a) (b)

图 5.5 pIC_{50} 预测结果与真实值对比.(a)训练集(b)测试集

由图 5.5 可以看出，该图左侧为训练集的预测值与真实值的比较，右侧为测试集的预测值与真实值的比较，从整体上看，右图和左图存在一定差异，其是由于两图的样本容量密度不同，左图样本量较大，右图样本量相对较小。若从表 5.1 中 MSE 来看，对于训练集，近似 1400 个样本的预测值和真实值的均方误差为 0.0818；对于测试集，近似 600 个样本的预测值和真实值的均方误差为 0.5297，这样来看，随机森林预测模型所预测出的值与真实值是相近的，说明该模型具有较好预测能力，且可较真实的反应化合物活性值。

通过以上对五种机器学习模型的测试性能评估分析，本文决定选择随机森林来做最终的化合物活性预测模型。

5.4 IC50 值、pIC50 值的预测

直接利用上节所确定的随机森林预测模型对文件“ERα_activity.xlsx”的 test 表中 50 个化合物的 IC_{50} 值和对应的 pIC_{50} 值进行预测，预测结果直接利用 python 导入文件“ERα_activity.xlsx”的 test 表中的 IC_{50}_nM 列及对应的 pIC_{50} 列。

6 ADMET 性质分类预测模型

6.1 问题三分析

本问题要求利用文件“Molecular_Descriptor.xlsx”和文件“ADMET.xlsx”所提供的分子描述符和 ADMET 数据，分别建立 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型，并用所建立的分类预测模型对文件“ADMET.xlsx”的 test 表中的 50 个化合物进行预测。通过观察文件“ADMET.xlsx”的 train 表中的 ADMET 数据发现，本问题需建立二分类预测模型。难点在于，如何建立适合本问题数据结构的、分类预测效果较好的二分类预测模型。

经分析本问题有三种潜在的解决方案：

方案 1：使用机器学习方法分别训练关于 ADMET 的 5 个性质的 5 个分类器；

方案 2：此问题同一输入数据，在 5 个类上输出不同的分类结果，是典型的多标签分类问题。我们可以采用相关的机器学习（sk-learn 库中支持多标签分类）算法，训练一个同时预测 5 性质的分类模型（通过单个样本一次性预测 ADMET 五个性质的情况）；

方案 3：问题同样适合采用深度学习的“多任务学习方法”解决，采用一维卷积构建神经网络，详细结构见图 6.1。

关于上诉三个方案，我们做了初步试验，结果如下：

表 6.1 多标签分类和二分类的对比

模型	属性	Acc	Pre	Rec	F1_score
多层感知机	多标签	0.527825	0.859586	0.850316	0.854558
K 近邻	多标签	0.556492	0.859191	0.85426	0.856554
随机森林	多标签	0.632378	0.882158	0.875124	0.878113
决策树	多标签	0.458685	0.802247	0.763995	0.777724
随机森林（Caco-2）	二分类	0.917722	0.915102	0.911602	0.913269

如表 6.1 所示，我们先尝试方案 2，然后和方案 1 对比。我们先测试 4 种多标签分类方法在 ADMET 上的 5 性质分类的性能，然后测试二分类方法在 ADMET 性质中的分类性能。从表 6.1 中可以看出，多标签分类方法的精度、召回率、F1_score、精确度均不如二分类方法。因此对方案 2 选择弃用，方案 3 的结构图如下。

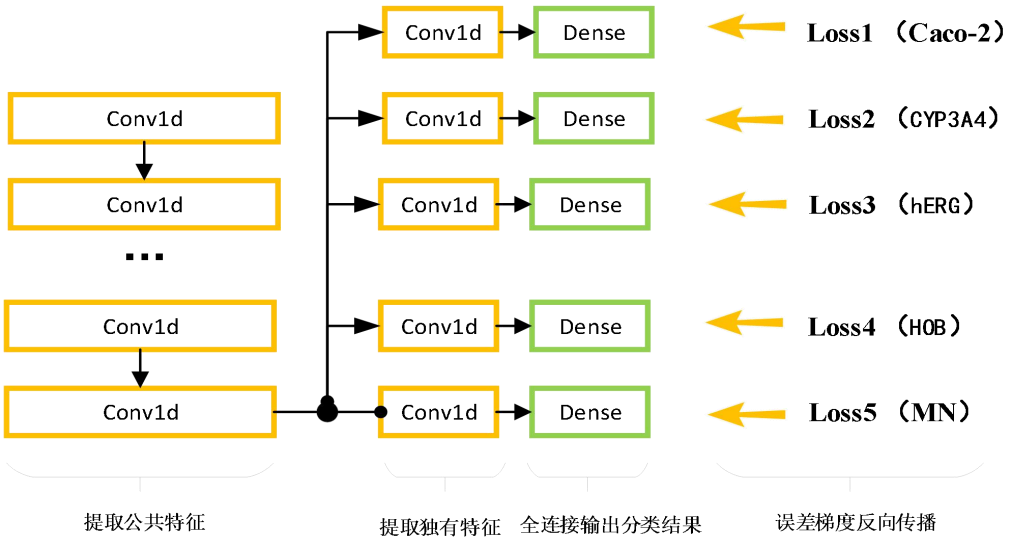


图 6.1 一维卷积神经网络多任务学习结构图

在方案 3 中，打算构建两个阶段的卷积神经网络分别提取五个分类任务的公共特征和每个分类任务的独有特征。网络结构图 6.1 所示，先用若干层一维卷积提取 ADMET 五个性质的共有特征，然后使用 5 个分支分别提取每个性质的独有特征进行二分类任务。在深度卷积神经网络的训练过程种，每个分支根据各自的 loss 函数将梯度传回每个分支，进而回传到提取公共特征的部分。5 个分支回传的梯度汇集在一起，存在相互引导的关系。这样使得第一部分一维卷积网络提取到的特征更加具有泛化性。通过观察发现，ADMET 五个性质的样本数据分布不均衡，正负类呈现近似 1：3 的分布，五个 loss 函数相互牵制在一定程度能克服数据分布不均带来的过拟合问题。

经过上述分析，方案 1 和方案 3 作为本问题潜在的解决方法，考虑到方案 3 存在一定的不确定性，故将方案 1 作为最终敲定的方案，把方案 3 作为可能的创新方案，在文章的第 8 部分“模型的评价与改进”给出了推广建议。

就方案 1 而言，使用机器学习方法训练 5 个分类器分别对 ADMET 的 5 个性质进行二分类。从所提供的数据本身出发，发现数据是具有一定非线性的，故本文决定：首先，对文件“Molecular_Descriptor.xlsx”和文件“ADMET.xlsx”中所提供的数据进行标准化处理，采用数据中随机抽取的 70%数据用于多种机器学习模型训练，使用剩下的 30%数据对模型进行验证和评价；其次，利用多种适合非线性数据的二分类机器学习方法来进行初次建模；然后，利用分类模型评价指标对所建立的多种预测模型进行性能测试；最后，针对化合物的 ADMET 性质：Caco-2、CYP3A4、hERG、HOB、MN，分别选择与其对应性能最优的分类预测模型。问题三的思路流程图如图 6.1 所示：

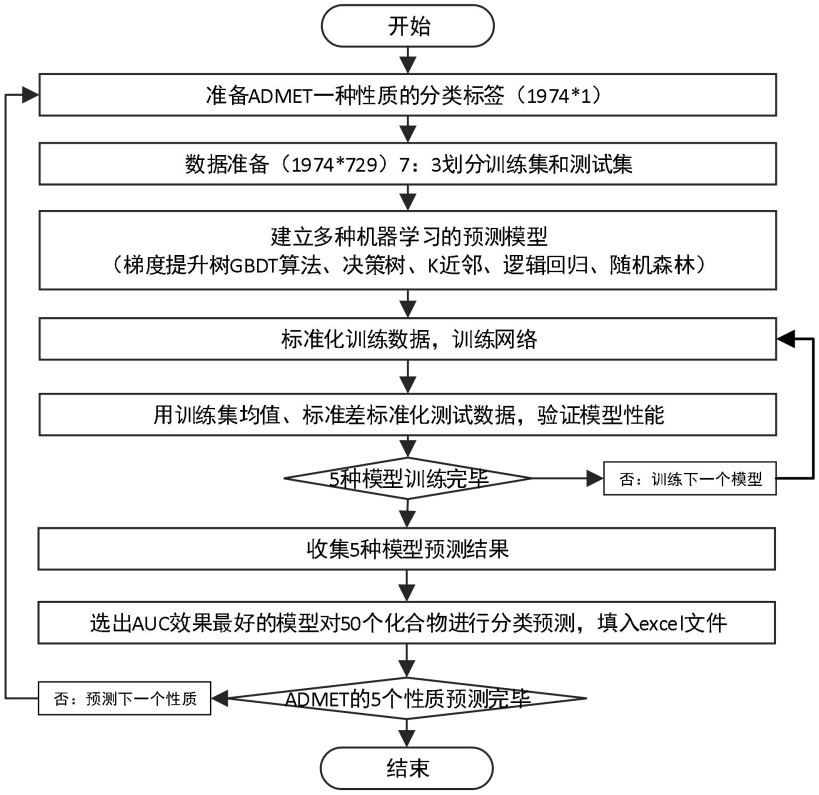


图 6.2 问题三思路流程图

6.2 ADMET 性质分类预测模型

本问题主要基于机器学习方法对 ERα拮抗剂的化合物 ADMET 性质构建二分类预测模型，其中，使用到多种机器学习分类方法，分别是：逻辑回归、K 近邻分类、随机森林、梯度提升树（GBDT 算法）、决策树。

6.2.1 多种机器学习分类预测模型基本原理

一．决策树

决策树（Decision Tree）是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。

总的来说，决策树就是树的类似结构，其树中的每个节点表示对一个数据样本特征属性的测试，而每个分支则表示测试后节点的输出，每个叶点表示不同的分类结果。由于决策树包括分类树和回归树，所以该模型可以用于分类和回归，本文选用决策树中的分类树来进行建模。

二．逻辑回归

逻辑（Logistic）回归是一种经常用于解决二分类（0 or 1）问题（也可以解决多分类问题）的机器学习方法，且在很多分类任务中的表现效果都很不错。Logistic 回归主要是通过使用其自身所固有的 logistic 函数来进行概率估算，从而来判断因变量（我们想要预测的标签）与一个或多个变量（特征）之间的关系。然后这些估算出来的概率结果必须二值化才能对样本类型进行预测。这就是 logistic 函数的任务，也将该函数称作 sigmoid 函数。

Sigmoid 函数的形状呈现的是一个 S 形曲线，它可以把任何实数值映射到区间(0,1)之间，但并不会取到 0 或者 1。然后利用一个阈值分类器把所得的 0 和 1 之间的值转换为不是 0 就是 1。由于 Logistic 回归是非线性模型，故通常使用极大似然估计对 Logistic 回归模型中的参数进行估计。使用该模型时，需要样本数据满足以下假设条件：

（1）线性敏感，自变量之间是非线性关系；

（2）残差和因变量都要服从二项分布。

三．随机森林

1、随机森林分类主要思想

随机森林算法在问题一中已有介绍，但在问题一、问题二中都是利用随机森林进行特征选择和数据回归，在本问中，利用随机森林对数据进行分类。该模型在分类的应用中，一般采用自助重采样的方法，从原始数据集中反复随机选取 n 个样本，生成一个新训练样本集来训练决策树，然后重复上述步骤 m 次，最终便可以生成 m 个决策树，形成一个随机森林，新数据的分类结果则是按照由分类树投票多少所形成的分数而定。

2、随机森林分类大致过程如下：

（1）从原始样本数据集中有放回的，随机采样得出 n 个样本；

（2）从所有的特征中随机选择 k 个特征，对步骤（1）所选出的样本使用这些特征建立决策树（通常情况下是 CART）；

（3）将步骤（1）（2）重复 m 次，即生成 m 棵决策树，继而组成随机森林；

（4）对于新数据，经过每棵树决策，最后投票确认分到哪一类。

四．K 近邻分类

KNN（K- Nearest Neighbor）法即 K 最邻近法，最早是同 Cover 和 Hart 在 1968 年提出的，它不仅是一个理论上比较成熟的方法，也还是机器学习算法中最简单方法模型之一[6]。该方法的思路在理解上较为简单直观：如果某样本特征空间中 K 个最相似（即最相近）的样本中，大多数样本都归属于某一个类别，那么这个样本也被划分在这个类别中。KNN 只是依照最近的一个或多个样本的类别来确定所要划分的样本类别。

该算法计算流程如下：

1）计算未被分类点和已知类别点之间的距离远近值；

2）遵照距离大小递增的次序进行排序；

3）选则与未被分类点距离最小的前 K 个样本点；

4）确定步骤（3）所选出的 K 个点所在类别的出现频数；

5）根据前 K 个点中出现频率最高的类别作为未被分类点的预判类。

其中，KNN 算法在使用的过程中需要注意以下几个关键点：首先，超参数 K 的大小对算法结果影响较大，不易把握；其次是距离的度量方法，需要根据数据样本本身的分布特点，选取合适的度量方法；最后，该算法的分类决策规则是按照少数服从多数，哪个类别多就判给哪个类；综合上述三点，K 值的选取和距离度量方法的设置是模型使用中最应关注的问题。

五．梯度提升树

梯度提升树算法（Gradient Boosting Regression Tree，GBDT）由多元回归树模型组成，使用了前向分布算法。前向分布算法的最终目的是为了最小化模型损失函数，并依照当前建立的模型及其模型的拟合函数来选取更为合适的决策树函数。通俗来说，GBRT 的基本思想就是通过计算每个树为预测前面树的残差而建立的简单回归树的序列。GBDT 主要是由两部分组成：

（1）决策树（Decision Tree，DT）：GBDT 是迭代的决策数算法，一般分为两种决策树——回归树和分类树，在 GBDT 中使用回归树，无论是在回归中应用还是分类中应用，因为

该算法每次迭代需要拟合的值是梯度值，该值是连续值，从而使用回归树更为合适。
（2）梯度提升（Gradient Boosting，GB）：该算法在迭代过程中需拟合梯度值，其核心是保证所学的每一棵树都是该模型学习之前所有树的结论和残差。

6.2.2 化合物 ADMET 性质分类预测模型建立

在化合物 ADMET 性质分类预测模型的建立中，本文主要采用上述所介绍的五种机器学习方法：逻辑回归、K 近邻分类、随机森林、梯度提升树、决策树。本问题首先利用文件“Molecular_Descriptor.xlsx”和文件“ADMET.xlsx”中 train 表中数据对这五种机器学习方法模型进行训练和测试；其次，将五种机器学习分类模型的测试结果进行比较评估；再次，针对化合物的 ADMET 性质：Caco-2、CYP3A4、hERG、HOB、MN，分别选择与其对应性能最优的分类预测模型；最后，利用相对于每个 ADMET 性质分类预测效果最优的模型，对文件“ADMET.xlsx”中 test 表的各个 ADMET 性质进行分类预测，并将结果填入“ADMET.xlsx”的 test 表中对应的 Caco-2、CYP3A4、hERG、HOB、MN 列。

一．数据准备

此过程中需对文件“Molecular_Descriptor.xlsx”中的 1974 组数据和文件“ADMET.xlsx”中的 ADMET 性质数据进行整理，将整理后的数据集随机抽取 70%和 30%为模型的训练集（70%）和测试集（30%），然后用 python 对训练和测试模型所用到的数据进行标准化处理，标准化方式与问题二中所提到的数据标准化一致。

二．ADMET 分类预测模型建立

利用整理好的数据对上述所提到的五种机器学习分类预测模型进行训练测试，并根据以下分类评价指标选出模型测试效果性能最优的，整体流程框架如图 5.2.2.1 所示：

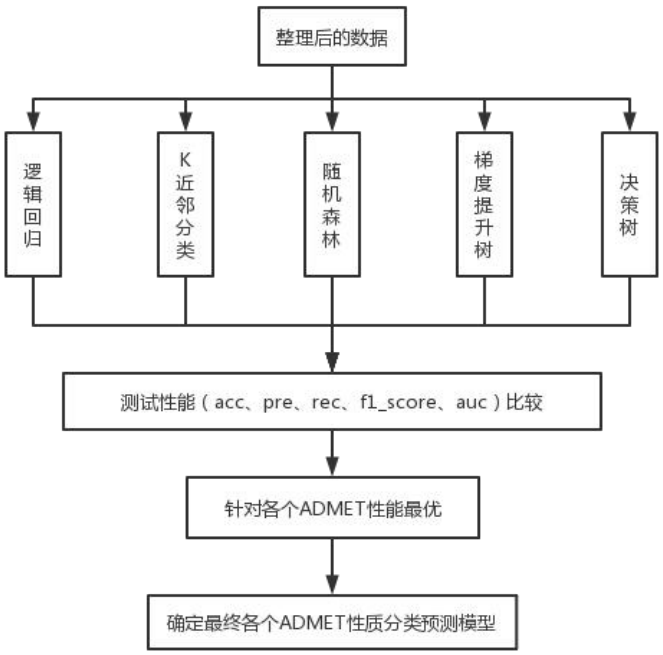


图 6.3 最优模型确立流程框架图

6.3 模型求解与结果分析

本文采用机器学习库 sk-learn 对上述五种分类模型进行实现，并采用 Acc、Rec、AUC 等作为分类模型性能评价指标，详情如下：

1、混淆矩阵[7]（Confuse Matrix）

针对一个二分类问题，即将实例分成正类（positive）或负类（negative），在实际分类中会出现以下四种情况：

（1）若一个实例是正类，并且被预测为正类，即为真正类 TP(True Positive)；

（2）若一个实例是正类，但是被预测为负类，即为假负类 FN(False Negative)；

（3）若一个实例是负类，但是被预测为正类，即为假正类 FP(False Positive)；

（4）若一个实例是负类，并且被预测为负类，即为真负类 TN(True Negative)。

混淆矩阵的每一行是样本的预测分类，每一列是样本的真实分类（反过来也可以），混淆矩阵具体形式如下：

真实标签 预测标签	正例	反例
正例	TP（真正类）	FN（假反类）
反例	FP（假正类）	TN（真反类）

1、准确率（Accuracy）：

该指标计算的是预测正确的样本数量占总样本数量的百分比，其值越大说明分类效果越好，计算公式如下[7]：

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}. \tag{6.1}$$

2、精准率（Precision）：

又称为查准率，是针对预测结果而言的一个评价指标。其含义是在模型预测为正样本的结果中，真正是正样本所占的百分比，该值越大则说明分类效果越好，计算公式如下[7]：

$$\Pr ecision = \frac{TP}{TP + FP}. \tag{6.2}$$

3、召回率（Recall）:

又称为查全率，是针对原始样本而言的一个评价指标。其含义是在实际为正样本中，被预测为正样本所占的百分比，该值越大则说明分类效果越好，具体计算公式如下[7]：

$$\mathrm{Re}\,call = \frac{TP}{TP + FN}. \tag{6.3}$$

4、F1 值（F1 score）：

为了能够评价不同算法优劣，在 Precision 和 Recall 的基础上提出了 F1 值的概念，来对 Precision 和 Recall 进行整体评价，F1 值越高模型的分类效果越好。具体计算公式如下[7]：

$$F1 = \frac{2 \times \Pr e \times \mathrm{Re}\,c}{\Pr e + \mathrm{Re}\,c}. \tag{6.4}$$

5、AUC（Area Under Curve）

AUC(Area Under Curve)是 ROC 曲线下方面积的大小[7]。AUC 值越大，代表模型的分类性能越好。综合上述的几个评价指标，在分类模型性能评价中，AUC 相较于 ACC，PRE，REC，F1_score 指标来说更具综合性，因此，在下面的分类预测性能评估中，本文在更加看重 AUC 值的好坏。

经过试验，将各个 ADMET 性质五个分类预测模型所获得的分类性能指标值进行汇总，详情如表 6.2：

表 6.2　ADMET 各性质对不同模型的性能指标

ADMET 性质	分类模型	Acc	Pre	Rec	F1_score	AUC
Caco-2	梯度提升树	0.9050	0.9008	0.8994	0.9001	**0.9714**
	随机森林	0.9177	0.9151	0.9116	0.9132	0.9712
	逻辑回归	0.8924	0.8861	0.8891	0.8875	0.9506
	k 近邻分类	0.8557	0.8610	0.8340	0.8433	0.9151
	决策树	0.8670	0.8611	0.8590	0.8600	0.8598
CYP3A4	随机森林	0.9379	0.9179	0.9193	0.9186	**0.9848**
	梯度提升树	0.9392	0.9213	0.9185	0.9199	0.9843
	逻辑回归	0.9278	0.9047	0.9060	0.9053	0.9724
	k 近邻分类	0.8949	0.8526	0.9099	0.8732	0.9398
	决策树	0.9202	0.8857	0.9171	0.8994	0.9090
hERG	梯度提升树	0.8949	0.8957	0.8910	0.8929	**0.9544**
	随机森林	0.8772	0.8772	0.8734	0.8750	0.9527
	逻辑回归	0.8582	0.8568	0.8555	0.8561	0.9203
	k 近邻分类	0.8342	0.8351	0.8394	0.8338	0.8820
	决策树	0.8430	0.8427	0.8383	0.8400	0.8771
HOB	梯度提升树	0.8696	0.8303	0.8179	0.8238	**0.9311**
	随机森林	0.8785	0.8414	0.8321	0.8366	0.9281
	逻辑回归	0.8481	0.7974	0.8052	0.8011	0.8881
	k 近邻分类	0.8165	0.7789	0.6907	0.7152	0.8106
	决策树	0.8430	0.7913	0.7951	0.7931	0.7979
MN	随机森林	0.9570	0.9661	0.9159	0.9379	**0.9897**
	梯度提升树	0.9532	0.9366	0.9350	0.9358	0.9771
	逻辑回归	0.9165	0.8867	0.8839	0.8852	0.9622
	k 近邻分类	0.8899	0.8408	0.9095	0.8643	0.9513
	决策树	0.9013	0.8821	0.8379	0.8569	0.8440

通过上表 6.2 可以看出，对于 ADMET 性质中不同性质，所得到的各个分类预测模型的性能指标也是具有一定差异的。对于 ADMET 性质中 Caco-2，hERG，HOB 这三个性质，我们所建立的五个初步分类预测模型中，梯度提升树的分类预测效果最好；对于 ADMET 性质中其他两个性质 CYP3A4，MN，随机森林的分类预测效果最好。

同时，本文还针对 ADMET 性质的每个性质所对应的最优分类预测模型，做了与之相对应的混淆矩阵，从而更加直观的看出针对不同性质所选出的最优分类预测模型的预测效

果，详情如下图所示：

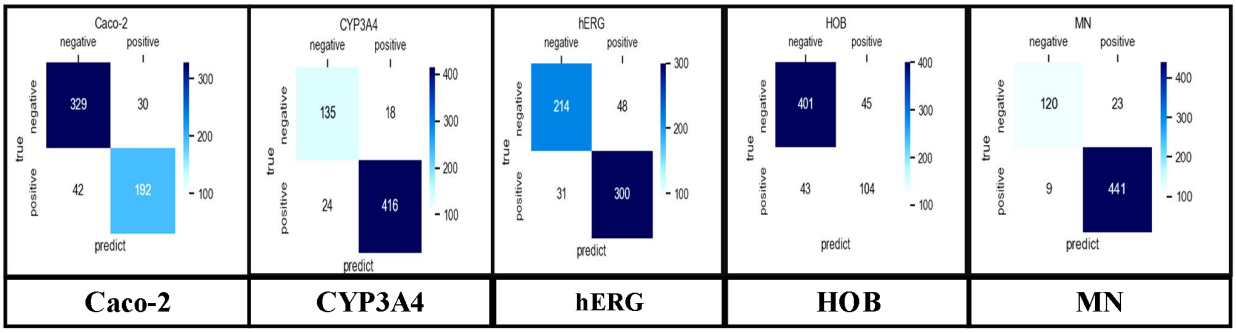


图 6.4 各个 ADMET 性质所对应最优分类预测模型的混淆矩阵

通过图 6.4 可以看出，通过对多种分类预测模型进行实验及性能比较，针对不同的 ADMET 性质所选出的最终分类预测模型的预测效果很好，对于每个 ADMET 性质，错分个数较少，分类正确样本数高达总样本数的 90%以上，可见，最终建立的分类预测模型效果很好。

因此在接下对文件“ADMET.xlsx”的 test 表中 ADMET 性质做分类预测时，会使用训练好的梯度提升树对以下三个性质做分类预测：Caco-2，hERG，HOB；使用训练好的随机森林对性质 CYP3A4，MN 做预测。

6.4 ADMET 五个性质分类预测

直接利用上节对不同 ADMET 性质所确定的最终分类预测模型，对文件“ADMET.xlsx”的 test 表中 50 个化合物进行相应的 ADMET 性质预测，预测结果直接利用 Python 导入文件文件“ADMET.xlsx”的 test 表中性质 Caco-2、CYP3A4、hERG、HOB、MN 的对应列。

7 ERα拮抗剂的优化建模

7.1 问题四分析

本文要求对 20 个主要分子描述符号进行优化，寻找目标变量（pIC_{50}）在最优情况下分子描述符号的取值或者取值范围。

此问题可以抽象成约束条件下最优化目标的搜索问题，难点在于本问约束条件较多，不但约束自变量的取值（操作条件需要在限定范围内进行调整），而且还需要判断因变量是否满足题设条件（ADMET 的 5 个性质中 3 个较好）。因为选取的因变量连续，故不能通过穷举的方式选取最优解，另外自变量与因变量之间并无明显的函数关系，所以无法使用普通的函数求极值方法进行优化。经分析，本文决定采用粒子群算法获取全局最优解，另外粒子群算法有平行计算、共享信息、速度较快等优点，可在短时间内找到最优解及其对应的自变量取值。

在问题 1 和问题 2 中，我们得到了前 20 个对预测模型影响较大的分子描述符，同时也获得了高性能的回归预测模型；在问题 3 中我们取得了 Caco-2、CYP3A4、hERG、HOB、MN 共 5 个性质关于 729 个分子描述符的 5 个分类预测模型。我们可以将问题 2 与问题 3 中的 6 个模型嵌入粒子群算法的适应性函数，以此实现回归预测（预测 pIC_{50} 值）、条件判断（在 5 个分类器输出结果中是否至少三个性质较好）功能。在适应性函数中我们将少于 3 个性质较好的预测结果直接返回，相反，满足 3 个性质较好的预测结果，我们返回其相反数，详见图 7.1。

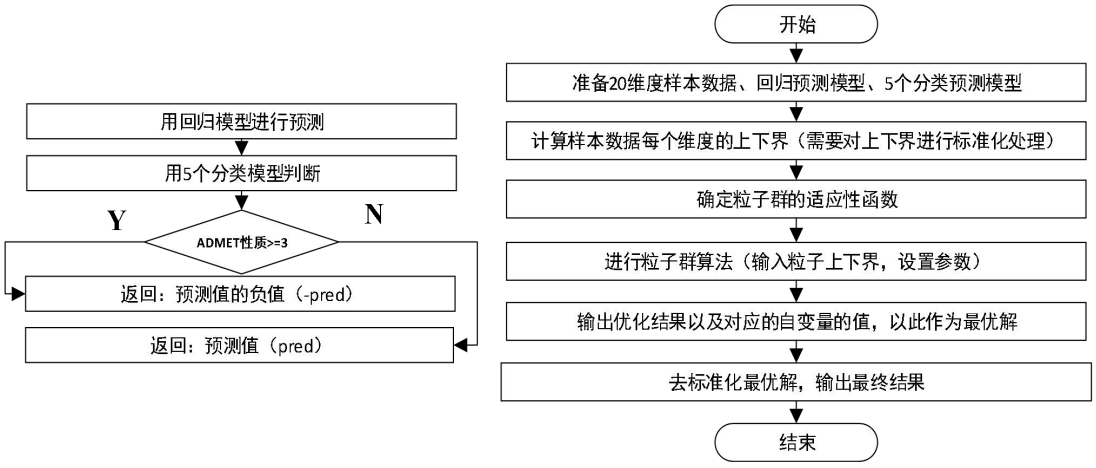


图 7.1 适应函数与算法流程图

如图 7.1，此处因粒子群算法朝着值变小的方向优化，满足条件时，返回预测值对应的负值，若小于当前记录的最小值记录，则更新算法最小值的记录，当不满足条件时，直接返回的正结果，在 PSO 内部参与最小值比较时不会更新任何记录（记录都是负值）；这样既不违背算法优化的方向，又能使预测值朝着增大的方向发展。此处我们不能直接使用问题 3 的分类器，为了与问题 2 的预测模型的输入格式保持一致，我们使用问题一中挑选的 20 个相对重要的分子描述符变量（将分类器输入特征的维度减少到 20 维），训练问题 4 中需要的 5 个分类器（因算法中粒子的维度必须与分子描述符的维度相同，所以此方法

也避免 729 维输入特征对例子群算法带来的计算负担）。

7.2 ERα拮抗剂的优化建模

7. 2. 1 粒子群算法原理

粒子群算法（Particle Swarm Optimization, PSO）通过模拟鸟群中的鸟来设计一种无质量的粒子，粒子包含两个属性：速度和位置，速度代表移动的快慢，位置代表当前的状态。每个粒子在搜索空间中独自的寻找最优解，寻找过程不断更新最优解并将其记为当前个体极值，然后将个体极值与粒子群里的其他粒子共享，找到最优的那个个体极值作为整个粒子群的当前全局最优解，粒子群中的所有粒子根据自己找到的当前个体极值和整个粒子群共享的当前全局最优解来调整自己的速度和位置。

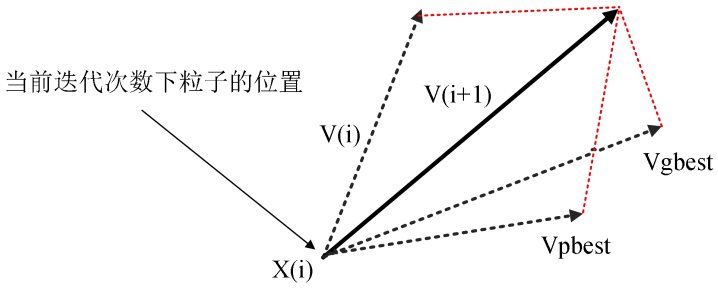


图 7.2 粒子更新原理

如图 7.2，PSO 开始时，初始化一群随机粒子(随机位置)。然后通过迭代、更新每个粒子的个体最优位置找到全局最优位置（最优解）。在每一次的迭代中，粒子通过跟踪两个“极值”(pbest，gbest：个体最优位置和全局最优位置)来更新自己。在找到这两个最优值后，粒子通过公式（7.1）和（7.2）来更新自己的速度和位置。

$$v_i = v_i + c_1 \times rand(0,1) \times (pbest_i - x_i) + c_2 \times rand(0,1) \times (gbest_i - x_i) \quad (7.1)$$

$$x_i = x_i + v_i \quad (7.2)$$

在公式（7.1）（7.2）中，$i = 1,2,....N$，N 是粒子的总数，v_i 是粒子的速度，c_i 是学习因子，通常 $c_1 = c_2 = 2$，$rand(0,1)$ 是 $(0,1)$ 之间的随机数。公式(7.1)的第一部分称为记忆项，表示上次速度大小和方向的影响；公式(7.1)的第二部分称为自身认知项，是从当前点指向粒子自身最好点的一个矢量，表示粒子的动作来源于自己经验的部分；公式(7.1)的第三部分称为群体认知项，是一个从当前点指向种群最好点的矢量，反映了粒子间的协同合作和知识共享。粒子就是通过自己的经验和同伴中最好的经验来决定下一步的运动。以上面两个公式为基础，形成了 PSO 的标准形式（7.3）：

$$v_i = \omega v_i + c_1 \times rand(0,1) \times (pbest_i - x_i) + c_2 \times rand(0,1) \times (gbest_i - x_i) \quad (7.3)$$

ω 称为惯性因子，其值非负。ω 较大时，全局搜索能力强局部搜索能力较弱，ω 较小时，全局搜索能力弱，局部搜索能力强。动态的 ω 在 PSO 算法中动态变化能获得比固定的 ω 更好的搜寻优结果。根据 PSO 算法的更新原理我们可以总结出粒子群算法的详细过程，如下所示:

1. 初始化粒子群（群体规模 N）：随机初始每个粒子的位置和速度；
2. 根据适应性函数，计算每个粒子的适应性指标；
3. 比较每个粒子的当前位置与个体最佳位置，若当前位置优于个体最佳位置，则用当前位置更新个体最佳位置；
4. 比较每个粒子的当前位置与全局最佳位置，若当前位置优于全局最佳位置，则用当前位置更新全局最佳位置；
5. 根据公式（7.2）、（7.3）更新每个粒子的速度与位置；
6. 若不满足结束条件，重复 2-5 步骤；否则返回全局最佳位置，算法结束。（结束条件一般为 max_iteration 或者适应性函数的迭代增量小于给定值。）

根据上述步骤，粒子群算法的伪代码如下表所示。

表 7.3 粒子群算法伪代码

Algorithm: PSO

```
Require: search range, population size N
  for each particle i
      Initialize velocity V(i) and position X(i) for each particle i
      evaluate particle i and set pBest(i) = X(i)
  end for
  gBest  = min{ pBest(i) }
  while not stop
      for i=1 to N
          Update the velocity and position of particle i
          Evaluate particle i
          if fit[x(i)] < fit[pBset(i)]
              pBest(i) = X(i);
          if fit[pBset(i)] < fit[gBset(i)]
              gBest(i) = pBest(i);
      end for
  end while
  print gBest
end procedure
```

7.2.2 优化目标及约束设定

（1）决策变量：

本文所建立的模型中影响目标变量（pIC_{50}）的主要变量一共有 20 个，因此该粒子群算法模型中共有 20 个可变变量，即这里的决策变量有 20 个，记为：

$$X = \{x_1, x_2, x_3 ... x_{20}\} \quad (7.4)$$

（2）目标函数：

问题要求在 3 个 ADMET 性质较好前提下，寻找能够使化合物对抑制 ERα具有更好的生物活性的分子描述符的值或取值范围。这里我们以目标变量（pIC_{50}）的值为主要的优化

目标，要求 pIC_{50} 的值尽可能大（- pIC_{50} 尽可能小），从而保证优化后结果对抑制 ERα具有更好的生物活性。

$$\max : regress\_predict(X) \tag{7.5}$$

（3）约束条件：

对于该优化问题，由于提出的 20 个决策变量在实际操作中存在一定的取值范围，因此存在约束 1：

$$nor(\min(x_i)) < x_i < nor(\max(x_i)) \tag{7.6}$$

$nor()$ 函数为标准化处理，减去均值除以标准差，使其满足数据均值 0、方差为 1，因为预测模型与分类模型都的训练数据都进行了标准化处理，所以此处的上下界也应该用训练集的均值和标准差进行标准化处理。此外，考虑到满足至少 3 个 ADMET 性质较好，因此有约束 2：

$$sum(classify\_predict(X) == [1,1,0,1,0]) >= 3 \tag{7.7}$$

其中列表“$[1,1,0,1,0]$”的每个数字代表 ADMET 的一个较好的性质，分别为 Caco-2：‘1’代表该化合物的小肠上皮细胞渗透性较好；CYP3A4：‘1’代表该化合物能够被 CYP3A4 代谢；hERG：‘0’代表该化合物不具有心脏毒性；HOB：‘1’代表该化合物的口服生物利用度较好；MN:‘0’代表该化合物不具有遗传毒性。只有当 $classify\_predict(X)$ 的预测序列中有三个元素与之对应位置元素相同，$regress\_predict \bmod el(X)$ 的结果才会参与算法内部的最优结果比较。

综上，约束条件为：

$$s.t.\begin{cases} nor(\min(x_i)) < x_i < nor(\max(x_i)) : i = 1,2,3,\cdots,N \\ sum(classify\_predict(X) == [1,1,0,1,0]) >= 3 \end{cases} \tag{7.8}$$

7.2.3 模型参数设定

依据题目要求，选用局部粒子群算法，结合相关因素和多次调试，确定粒子群算法的主要变量设定值如表 7.1 所示。

表 7.1 粒子群算法要素设定

要素名称	符号	值
种群大小	P	50
维度	D	20
权重因子	ω	0.8
学习因子	c	2
最大迭代次数	t	100
初始化粒子的位置	x	随机初始化
初始化各种粒子的速度	v	随机初始化

7.3 模型求解及结果分析

针对上述建立的复杂多约束优化模型，本文使用粒子群算法对模型进行求解，并致力于找到最大的 pIC_{50} 对应的分子描述符号的取值。

首先，分类模型和回归模型均随机采用 70%数据的均值和标准差对数据进行标准化(与问题二标准化一致)。虽然输入数据来自同一数据集，但无法保证，两个模型标准化过程采用的均值和标准差相同。并且我们采用每个分子描述符所有数据的均值和标准差，对每个数据的上界和下界进行标准化。若三个过程标准差各不相同，则会影响模型的预测和分类的性能。我们提取问题 1 中 20 个分子描述符 70%的数据，比较其中 9 个分子描述符的部分指标与整体指标的差距。详见表 7.2

表 7.2 部分数据与整体数据的均值与标准差比较

分子描述符	总体均值	均值（70%data）	总体标准差	标准（70%data）
MDEC-23	25.58675	25.67847	9.463304	9.554357
minsssN	0.95112	0.962398	1.102871	1.10215
C1SP2	0.891084	0.888406	1.301465	1.332649
maxssO	3.121853	3.164688	2.979785	2.977172
BCUTc-1h	0.206067	0.205272	0.085619	0.08445
BCUTp-1h	11.97345	11.96566	1.327026	1.328269
minHBa	2.823383	2.802437	3.329286	3.322204
VCH-5	0.040646	0.040489	0.04998	0.048891
WTPT-4	8.735564	8.779146	4.375231	4.360688

根据表 7.2，我们可以看到 70%数据的指标与整体指标基本相同，因此我们假设上述三个标准化处理过程所用的均值与标准差相同。然后用随机森林算法对 Caco-2、CYP3A4、hERG、HOB、MN 五个指标训练出 5 个分类器，其中测试集的比例为 30%，训练集为 70%，每个分类器在对应数据测试集上的性能如表 7.3 所示：

表 7.3 ADMET 的 5 个分类器的性能

metrics	Acc	Pre	Rec	F1_score	AUC
Caco-2	0.9106	0.912	0.9001	0.9052	0.9729
CYP3A4	0.9292	0.9038	0.9139	0.9087	0.9803
hERG	0.882	0.8832	0.8772	0.8795	0.9527
HOB	0.8685	0.8267	0.8145	0.8203	0.9161
MN	0.946	0.9403	0.9096	0.9237	0.9794

通过表 7.3 可以看出，用随机森林对 ADMET 的 5 个性质进行分类都取得了较好的测试效果（AUC 均大于 90%），因此用上述五个分类器对粒子群算法产生的自变量的取值进

行预测具有较高的可靠性。

第二步，考虑到在适应性函数里强行屏蔽不满足条件的预测值会破坏搜索算法的连续性，影响搜索算法的优化方向。我们初次采用无判决条件限制（取消满足 3 个 ADMET 性质较好的限制）的适应性函数进行搜索。并用得到的结果进行预测。模型迭代曲线如图 7.4 所示，预测结果如表 7.4 所示。

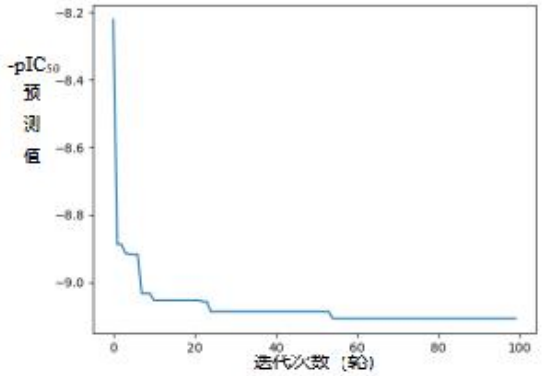


图 7.4 无 ADMET 性质限制的迭代优化曲线

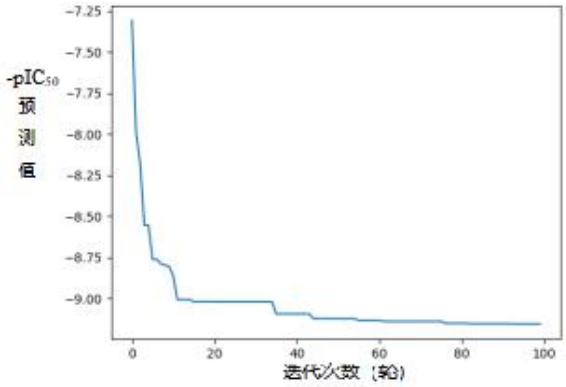


图 7.5 有 ADMET 性质数量限制的迭代优化曲线

通过上图可看出，算法在 50 轮迭代之后就已经平稳收敛，粒子群返回的 pIC_{50} 最优值为 9.10676421，但是只使得 ADMET 中 5 个性质中 2 个较好，不满足题设条件（详见表 7.4，其中半约束条件为不考虑 ADMET 性质满足情况的预测）。

因此我们在适应性函数里面嵌入“ADMET 五个性质中三个较好”的限制，如此一来，算法在运行的过程中，将不满足上述限制的粒子（对应 20 个分子描述符的取值）适应性函数值设为正值，正值则不会参与粒子群算法内部个体适应性函数值最低记录和全局适应性函数值最低记录的更新，因为粒子群算法朝着适应性函数值更小的方向优化，故其对算法寻优无影响。满足上述限制的粒子的适应性函数值为负值（适应性函数结构内部设计，满足条件则返回预测值的相反数），若其适应性函数值小于个体最低记录，则用该适应性函数值更新该个体最低记录；若其适应性函数值小于全局最低记录，则用该适应性函数值更新全局最低记录。在不停的迭代过程中，将持续更新个体最低记录和全局最低记录，当满足算法结束条件时，停止迭代，输出全局最低记录和对应的粒子的取值。

对加上 ADMET 性质约束后适应性函数，再次进行算法寻优，过程展示如图 7.6 所示，优化迭代曲线如图 7.5 所示。

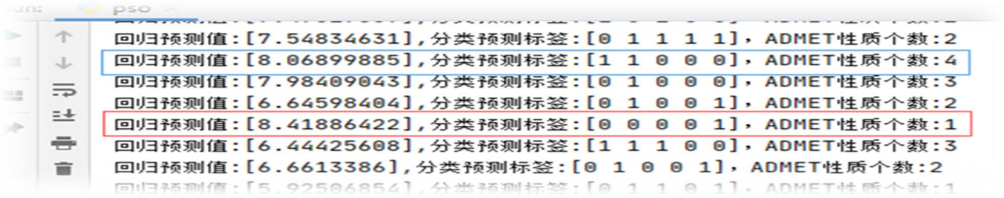


图 7.6 迭代过程展示

从图 7.5 中可以看到，ADMET 性质约束，没有影响 PSO 算法的搜索方向和收敛速度，第 10 轮迭代时已经将 pIC_{50} 预测值优化到 9（不考虑 ADMET 约束条件在相同条件下第 10 轮大约为 9），这也说明 ADMET 性质与 pIC_{50} 提升不存在冲突，可兼得。表 7.4 中“全约束下最优解”即为第二次搜索的最优解。

表 7.4 PSO 算法寻优结果

分子表述符	100 轮半约束下最优解	100 轮全约束下最优解	去标准化
MDEC-23	3.016373987	3.016373987	54.13161454
maxHsOH	1.755215349	0.287919598	0.498250929
minsssN	1.630702251	1.630702251	2.749574389
C1SP2	-0.665321377	13.97705028	19.0817255
maxssO	1.209433015	1.209433015	6.725703998
BCUTc-1l	-2.442987112	5.725798379	-0.18428731
minHBint5	1.582660791	1.58536316	3.147378159
VC-5	8.729961365	8.729961365	1.502156481
ATSc3	5.668701014	-3.580872009	-0.374342989
BCUTc-1h	-1.061907373	-1.568021419	0.071814467
MDEC-33	-2.231952452	8.937596431	49.12875266
SHBint10	16.12193655	14.68749508	99.64003475
XLogP	-4.018529298	-4.018529298	-3.524251538
mindssC	3.596836696	3.596836696	1.990345782
ETA_Shape_Y	3.497002231	3.497002231	0.562536072
BCUTp-1h	-3.042566916	3.620711027	16.77822835
minHBa	-1.622187513	-0.516956907	1.102286002
ATSc2	1.332774294	1.332774294	-0.02303766
VCH-5	9.055583665	9.055583665	0.493241994
WTPT-4	9.517526772	-1.994484906	0.009230742
回归预测值	**9.10676421**	**9.15606568**	**9.15606568**
ADMET 最佳标签	**[1 1 0 1 0]**	**[1 1 0 1 0]**	**[1 1 0 1 0]**
分类预测结果	**[0 1 0 0 1]**	**[1 1 1 1 1]**	**[1 1 1 1 1]**
ADMET 性质个数	**2**	**3**	**3**

从表 7.4 可以看输出，第二次搜索的最优解优于第一次搜索的，这也印证了我们的猜想，可以在取得更高的 pIC_{50} 的同时满足更多的 ADMET 性质。

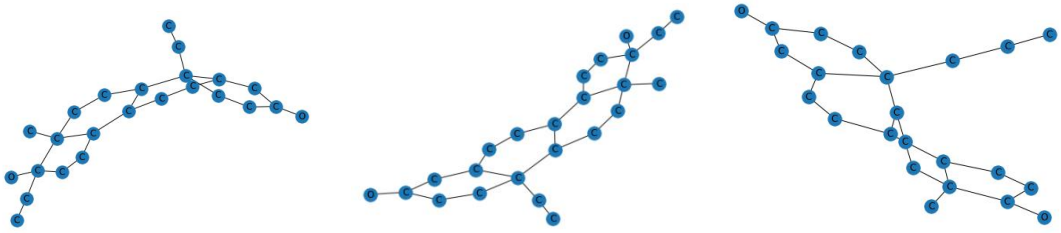
最后上述解均为标准化下的解，我们需要根据每个变量的均值和方差还原最优解对应的原始取值。根据上述对标准化过程的假设，此处采用完整数据的均值与方差对最优解进行还原。最优解的最终结果为表 7.4 中的“去标准化”列。

本文考虑到需要现实情况进行对比，因此在本题提供的 1974 个训练集化合物中搜寻满足全部 ADMET 性质的化合物的 pIC_{50} 数值进行比较。通过筛选在训练集化合物中选出了 11 个满足条件的目标，下表展示了其 SMILES 式和对应的 pIC_{50} 值。

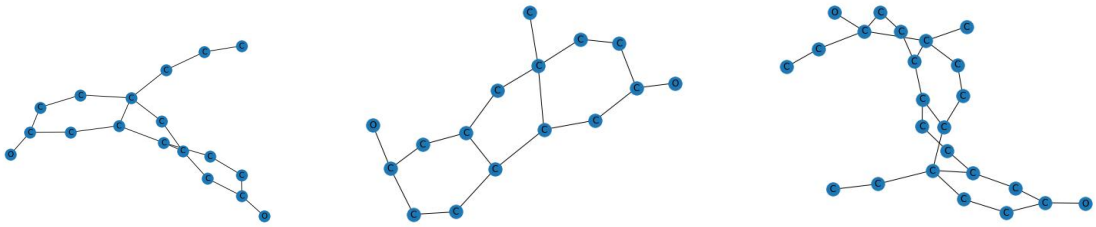
表 7.5　11 个满足全部 ADMET 性质的化合物及 pIC_{50} 数值

序号	SMILES 式	pIC_{50} 数值
1	C[C@]12CCC3C(CC=C4C[C@@H](O)CC[C@]34C=C)C1CC[C@@]2(O)C#C	6.363
2	C[C@]1(O)CCC2C3CC=C4C[C@@H](O)CC[C@]4(C=C)C3CC[C@]12C	5.978
3	CCC[C@]12CC[C@H](O)CC1=CCC3C4CC[C@H](O)[C@@]4(C)CCC23	5.288
4	CCCC12CCC(=O)C=C1c3ccc(O)cc3C2	5.427
5	CC12CCC(=O)C=C1c3ccc(O)cc3C2	5.190
6	C[C@]12CC[C@H]3[C@@H](CC=C4C[C@@H](O)CC[C@]34C=C)[C@@H]1CC[C@@]2(O)C#C	6.824
7	C[C@]1(O)CC[C@H]2[C@@H]3CC=C4C[C@@H](O)CC[C@]4(C=C)[C@H]3CC[C@]12C	5.979
8	CCOC(=O)C1=C(C)S\C(=C/N(C)C)\C1=O	4.395
9	C[C@]12CC[C@H]3[C@@H](CCC4=CC(=O)CC[C@H]34)[C@@H]1CC[C@@]2(O)C#C	6.093
10	C[C@]1(O)CC[C@H]2[C@@H]3CC[C@H]4Cc5n[nH]cc5C[C@]4(C)[C@H]3CC[C@]12C	6.947
11	C[C@@H]1CC2=C(CCC(=O)C2)[C@H]3CC[C@@]4(C)[C@@H](CC[C@@]4(O)C#C)[C@H]13	6.363

由该表格可发现在满足 ADMET 五个性质的化合物其 pIC_{50} 数值较小，符合在约束条件增加的情况下求解最优值变差的理论情况，也间接佐证了本文求解情况的存在可能性。下图是通过 pysmiles 库绘制的前六个化合物的分子结构图，未来可以通过结合实际分子结构图进行深入的研究。



(a)　(b)　(c)



(d) (e) (f)

图 7.7 六种满足 ADMET 性质的化合物分子结构图.
(a) 序号 1 化合物 (b)序号 2 化合物 (c)序号 3 化合物
(d)序号 4 化合物 (e)序号 5 化合物 (f)序号 6 化合物

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非线性和高耦合性，使用了随机森林回归等适用于处理非线性、高耦合性特征的方法。所获得的主要变量物理意义明确，符合实际。
(2) 本文在第二、三问中利用多种机器学习算法模型进行回归及二分类问题的求解，实现了在横向上解决方案的比较。最终在对比中发现最佳模型为随机森林，效果优异。
(3) 本文在第四问中创新采用了粒子群智能搜索算法实现了对于分子描述符对应最优生物活性的取值求解，充分考虑了变量范围等方面的约束，极大地提高了研发新药的效率。

8.2 模型缺点

(1) 本文训练数据有限，后期可获取更多数据优化模型。
(2) 本文所选用的机器学习算法的参数选择还具有一定的主观性。
(3) 在问题二，问题三中仅使用了经典的机器学习模型，在深度神经网络这方面，仅涉及 MLP 模型，该模型网络深度较浅，未建立更深层次的网络模型进行尝试。
(4) 对于真实医药研发模型的学习较少，可能对于问题专业性的考虑还不足。

8.3 模型的改进与推广

8.3.1 模型改进

1．ADMET 的五个性质分类中我们选取 AUC 作为关键指标，现实情况下，对于“小肠上皮细胞渗透性较好”、“化合物能够被 CYP3A4 代谢”、“化合物的口服生物利用度较好”三个指标，在制备过程中，更关心它们的这些利好的真实性（如果错将 0 判断为 1 则这些化合物将会对最终结果产生负面影响），所以对于 Caco-2、CYP3A4、HOB 三个性质分类任务中，选择 Precision（精确度：所有预测为“性质较好”的样本里面真实“性质较好”的比例）指标作为最终评价指标更为合适。

2．对应的“化合物具有心脏毒性”、“化合物具有遗传毒性”两个性质，我们希望找到这样的有害的化合物（预测为阳性），所以对于 hERG、MN 两个性质的分类任务，使用 Recall（召回率：我们预测为“存在毒性”的化合物数量占真实“存在毒性”的化合物数量的比例）作为最终指标比较合适。

3．在粒子群算法的寻优过程中，只要求 ADMET 三个性质较好，真实的制备过程中，我们可以根据 ADMET 的五个性质的重要性，将五个性质排序，要求前三个重要的性质预测较好。以此筛选化合物，进一步减少对健康的危害。

4．本文中，我们没有采用深度学习的方法，进行分类预测。在实际环境下从部署和实现的角度衡量，深度学习更适合应用推广。将采取问题 3 的方案三，训练神经网络（尽可能解决数据平衡问题），将网络参数部署在服务器上。设值定期训练策略，让长期的制

备数据通过深度学习方法产生高性能的化合物性质判定模型。

8.3.2 模型推广

1．所采用的降维和预测方法适用性广，对后期各方面研究具有一定参考意义；
2．可以将本模型应用于其他癌症药物靶点的研究中。

9 参考文献

[1] 刘睿,赵静,张志,曹晓艳,杨照环. 雌激素受体亚型 ERα、ERβ在乳腺癌中的表达及意义[J]. 中国实验诊断学,16(06):1092-1094，2012.

[2] 李晓,李达,周雪松,赵勇.化合物 ADMET 性质预测平台的构建[J].生物信息学,15(03):179-185，2017.

[3] Murphy L C,Simon S L,Parkes A,Leygue E,Dotzlaw H,Snell L,Troup S,Adeyinka A,Watson P H. Altered expression of estrogen receptor coregulators during human breast tumorigenesis.[J]. Cancer research,60(22),2000.

[4] Breiman,L.(2001a).“Random Forests.”Machine Learning,45(1):5-32,2001.

[5] 南风几经秋雨.人工智能应用技术之机器学习（通识篇）.https://www.163.com/dy/article/GM6LAACK0552IANC.html.2021-10-15.

[6] 李景奎, 张义民. 基于 K 邻近算法的连续体结构拓扑优化设计[J]. 组合机床与自动化加工技术,(01):10-12,2012.

[7] 人工智能.分类算法评价指标详解.https://zhuanlan.zhihu.com/p/110015537.2021-10-16.

[8] Jshjdsjfdjsfdsf.Pearson’s product moment correlation/皮尔逊(森)积矩相关系数.https://blog.csdn.net/jshjdsjfdjsfdsf/article/details/39038739.2021-10-16.

[9] AIDD Pro. 药物分子设计的重要利器-QSAR.https://zhuanlan.zhihu.com/p/398324971.2021-10-17.

[10] Lawal Hadiza Abdulrahman,Uzairu Adamu,Uba Sani. QSAR, molecular docking studies, ligand-based design and pharmacokinetic analysis on Maternal Embryonic Leucine Zipper Kinase (MELK) inhibitors as potential anti-triple-negative breast cancer (MDA-MB-231 cell line) drug compounds[J]. Bulletin of the National Research Centre,45(1),2021.

[11] Stanton David T,Baker Jennifer R,McCluskey Adam,Paula Stefan. Development and interpretation of a QSAR model for in vitro breast cancer (MCF-7) cytotoxicity of 2-phenylacrylonitriles.[J]. Journal of computer-aided molecular design,35(5),2021.

[12] 冯怡. 血管抑制剂类抗癌药物的 QSAR 研究[D].陕西科技大学,2021.

10 附录

10.1 程序

问题 1：筛选出 20 个主要变量

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor

def initial_processing():
    #get raw data
    ER_alpha_path = r'C:\Users\HEMIN\Desktop\2021D\ERα_activity.xlsx'
    description_path = r'C:\Users\HEMIN\Desktop\2021D\Molecular_Descriptor.xlsx'
    ER_alpha = pd.read_excel(ER_alpha_path, sheet_name=None)
    description = pd.read_excel(description_path, sheet_name=None)
    ER_alpha = ER_alpha['training']
    description = description['training']
    return ER_alpha,description

def get_imf_rank(ER_alpha,description):

    target_c = ER_alpha.loc[:,'pIC50']
    factors = description.loc[:,'nAcid':]
    print(target_c.shape,factors.shape)

    factors_name = factors.columns.values

    x = factors.values
    y = target_c.values

    clf = RandomForestRegressor(n_estimators=300, random_state=(0), max_features=729, n_jobs=(2))
    clf.fit(x, y)
```

```
    #统计20个主要变量
    importances = clf.feature_importances_
    imf_list = []
    for index,imf in enumerate(importances):
        imf_list.append([factors_name[index],imf])

    df_tmp = pd.DataFrame(imf_list, columns=['features','importance'], dtype=float)
    df_tmp = df_tmp.sort_values(by="importance", inplace=False, ascending=False)

    print(df_tmp[:20])

    main_variable = factors.loc[:, df_tmp['features'][:20] ]
    main_variable.to_excel(r'./main_variable20.xlsx', index=False, header=True)
    target_c.to_excel(r'./target.xlsx', index=False, header=True)

if __name__ == '__main__':
    ER_alpha,description = initial_processing()
    get_imf_rank(ER_alpha, description)
```

问题 2：训练回归预测模型（8 种）

```
class MLRegressor():
    def __init__(self,x_path,y_path,nor_flag,test_size):
        tmp_x = pd.read_excel(x_path).values
        tmp_y = pd.read_excel(y_path).values

        self.x_train, self.x_test, self.y_train, self.y_test = train_test_split(
            tmp_x, tmp_y, test_size=test_size, shuffle=True, random_state=0)
        print(self.x_train.shape, self.x_test.shape, self.y_train.shape, self.y_test.shape)

        self.scaler = StandardScaler()
        if nor_flag:
            self.scaler.fit(self.x_train)
            self.x_train = self.scaler.transform(self.x_train)
            self.x_test  = self.scaler.transform(self.x_test )
        self.model = None

    def calPerformance(self,y_true, y_pred):
        """..."""
        dict = {}
        model_metrics_name = [explained_variance_score, mean_absolute_error, mean_squared_error, r2_score]
        for index,metric in enumerate(model_metrics_name):
            tmp_score = metric(y_true, y_pred)
            dict[metric] = tmp_score
        return dict

    def plot_curve(self,y_tre,y_pre,stage):...
```

```
    def get_model(self,model_name):
        if model_name == 'mlp':
            #hidden layer:100-50
            self.model = MLPRegressor(hidden_layer_sizes=(100,50), random_state=0, max_iter=1000, shuffle=True)
        if model_name == 'rf':
            #decision trsee:100
            self.model = RandomForestRegressor(n_estimators=100, random_state=(0), max_features=20, n_jobs=(2))
        if model_name == 'svr':
            #penalty coefficient :100
            self.model = SVR(kernel='rbf', C=100)
        if model_name == 'dtr':
            self.model = DecisionTreeRegressor()
        if model_name == 'linear':
            self.model = LinearRegression()
        if model_name == 'knn':
            self.model = KNeighborsRegressor(n_neighbors=1)
        if model_name == 'bag':
            self.model = BaggingRegressor()
        if model_name == 'gbrt':
            self.model = GradientBoostingRegressor(n_estimators=100,max_features=20)
```

```
    def train(self,is_show):

        self.model.fit(self.x_train, self.y_train.ravel())

        y_pred = self.model.predict(self.x_train)
        dict_tr = self.calPerformance(self.y_train, y_pred)
        tr_mse = mean_squared_error(self.y_train, y_pred)
        tr_scores = self.model.score(self.x_train, self.y_train)
        print('training model score: {:.4f},mse: {:.4f}'.format(tr_scores,tr_mse))
        if is_show:self.plot_curve(self.y_train, y_pred,stage='train')

        y_pred = self.model.predict(self.x_test)
        dict_te = self.calPerformance(self.y_test, y_pred)
        te_mse = mean_squared_error(self.y_test, y_pred)
        te_scores = self.model.score(self.x_test, self.y_test)
        print('testing  model score: {:.4f},mse:{:.4f}'.format(te_scores,te_mse))
        if is_show:self.plot_curve(self.y_test, y_pred,stage='test')

        return tr_mse,tr_scores,te_mse,te_scores,dict_tr,dict_te
```

```
    def pretict(self,op_up,sheetname):
        ER_alpha_path = r'C:\Users\HEMIN\Desktop\2021D\ERα_activity.xlsx'
        description_path = r'C:\Users\HEMIN\Desktop\2021D\Molecular_Descriptor.xlsx'
        ER_alpha = pd.read_excel(ER_alpha_path, sheet_name=None)
        description = pd.read_excel(description_path, sheet_name=None)
        ER_alpha = ER_alpha['test']
        description = description['test']

        var_names = pd.read_excel(r'./main_variable20.xlsx').columns.values
        x = description.loc[:, var_names].values

        y = self.model.predict(x)
        print(y)

        if op_up:
            ER_alpha['pIC50'] = y
        else:
            ER_alpha['IC50_nM'] = y

        book = load_workbook(r'C:\Users\HEMIN\Desktop\2021D\ERα_activity.xlsx')
        writer = pd.ExcelWriter(r'C:\Users\HEMIN\Desktop\2021D\ERα_activity.xlsx', engine='openpyxl')
        writer.book = book
        ER_alpha.to_excel(writer, index=False, header=True, sheet_name='answer{}'.format(sheetname))
        writer.save()
```

```
def run_all():
    test = MLRegressor(x_path = r'./main_variable20.xlsx',y_path = r'./target.xlsx',nor_flag=True,test_size=0.3)
    name_list = ['mlp','rf','svr','dtr','linear','knn','bag','gbrt']
    tmp_list = []
    for method in name_list:
        test.get_model(method)
        tr_mse,tr_scores,te_mse,te_scores,dict_tr,dict_te = test.train(is_show=False)
        #explained_variance_score, mean_absolute_error, mean_squared_error, r2_score
        tmp_list.append([method,tr_mse,tr_scores,
                         dict_tr[explained_variance_score], dict_tr[mean_absolute_error], dict_tr[r2_score],
                         te_mse,te_scores,
                         dict_te[explained_variance_score], dict_te[mean_absolute_error], dict_te[r2_score],
                         ])
    df = pd.DataFrame(tmp_list, columns=['method', 'train_mse','train_scores',
                                         'TR_EVS','TR_MAE','TR_R2S',
                                         'test_mse','test_scores',
                                         'TE_EVS', 'TE_MAE', 'TE_R2S'
                                         ])
    df = df.sort_values(by='test_mse',ascending=True,inplace=False).round(decimals=4)
    print(df)
    df.to_excel(r'./questionx.xlsx', index=False, header=True)
```

问题 3：分类预测模型（8 个）

```
class MLClassifier():
    def __init__(self,nor_flag,test_size,task_name,c5):...

    def calPerformance(self,y_tre, y_pre,y_score):...

    def print_confusion_matrix_1(self,y_tre, y_pre):...

    def plot_curve(self,y_tre,y_pre):...

    def get_model(self,model_name):
        if model_name == 'lgr':
            #hidden layer:100-50
            self.model = LogisticRegression(C=100.0,random_state=1)
        if model_name == 'svc':
            self.model = SVC(kernel='rbf',C=100,random_state=1,probability=True)
        if model_name == 'dtc':
            self.model = DecisionTreeClassifier(max_depth=5,max_features=20,random_state=1)
        if model_name == 'rfc':
            self.model = RandomForestClassifier(n_estimators=100,random_state=(0),n_jobs=2)
        if model_name == 'knn':
            self.model = KNeighborsClassifier(n_neighbors=2)
        if model_name == 'bag':
            self.model = BaggingClassifier()
        if model_name == 'gbdt':
            self.model = GradientBoostingClassifier(n_estimators=100)
        if model_name == 'guss':
            self.model = GaussianProcessClassifier(kernel)
```

```
def run_all(task_name):
    test = MLClassifier(nor_flag=True,test_size=0.3,task_name=task_name,c5=True)
    name_list = ['lgr','svc','dtc','rfc','knn','bag','gbdt']
    tmp_list = []
    for method in name_list:
        test.get_model(method)
        accuracy,precision,recall,F1_score,auc,scores,accuracy1,precision1,recall1,F1_score1,auc1,scores1 = test.train()
        tmp_list.append([method,accuracy,precision,recall,F1_score,auc,scores,accuracy1,precision1,recall1,F1_score1,auc1,scores1])
    df = pd.DataFrame(tmp_list,
                      columns=['method', 'train_acc','train_pre','train_rec','train_f1','train_auc','train_score',
                               'test_acc','test_pre','test_rec','test_f1','test_auc','test_score'])
    print(df.sort_values(by='test_auc',ascending=False))
```

```
def run_one(task_name,net):
    print('op_target:{}...'.format(task_name))
    sample = MLClassifier(nor_flag=True, test_size=0.3,task_name=task_name,c5=True)
    sample.get_model(net)
    sample.model.fit(sample.x_train, sample.y_train)
    y_pred = sample.model.predict(sample.x_test)
    y_score = sample.model.predict_proba(sample.x_test)
    accuracy,precision,recall,F1_score,auc = sample.calPerformance(sample.y_test,y_pred,y_score)
    print('accuracy:{:4f},precision:{:4f},recall:{:4f},F1_score:{:4f},auc:{:4f}'.format(
        accuracy, precision, recall, F1_score, auc))
    #sample.print_confusion_matrix_1(sample.y_test,y_pred)
    #sample.pretict(target_item=task_name,sheetname=task_name+net+'20variable',c5=True)
    #-----------------------------------------
    return sample.model, sample.scaler
```

问题 4：粒子群算法寻优

```
from ML_classifier import run_one

def pso_data_preaparation():
     factors = pd.read_excel(r'./main_variable20.xlsx',index_col=False,header=0)
     min_list = []
     max_list = []
     for col in factors.columns.values:
          tmp = factors[col].values
          min_list.append(np.min(tmp))
          max_list.append(np.max(tmp))
     return min_list,max_list,factors.columns.values

#min_list,max_list = pso_data_preaparation()

def get_five_classifier():
    c_list = ['Caco-2','CYP3A4','hERG','HOB','MN']
    c_model = {}
    c_scaler = {}
    for task_name in c_list:
        model, scaler =run_one(task_name=task_name, net='rfc')
        c_model[task_name] = model
        c_scaler[task_name] = scaler
    return c_model,c_scaler
```

```
from sko.PSO import PSO
#regressing
model,scaler = run_one()
min_list,max_list,var_list= pso_data_preaparation()
#normalize range
min_list = scaler.transform(np.array(min_list).reshape(1,-1))
max_list = scaler.transform(np.array(max_list).reshape(1,-1))
#classify preparation
c_model,c_scaler = get_five_classifier()
```

```
def judge(x,c_model):
    c_list = ['Caco-2', 'CYP3A4', 'hERG', 'HOB', 'MN']
    predict = []
    answer = np.array([1,1,0,1,0])
    for model_name in c_list:
        model = c_model[model_name]
        result = model.predict(x)
        predict.append(result[0])
    predict = np.array(predict)
    #print(predict)
    return sum((predict == answer)),predict
```

```
def demo_func1(x):
    x = [x]
    #scaler.transform(x)
    pred = model.predict(x)
    amount,pre_label = judge(x,c_model)
    print('回归预测值:{},分类预测标签:{}，ADMET性质个数:{}'.format(pred,pre_label,amount))
    if amount >= 3:
        return -pred
    else:
        return pred

def run_pso():
    pso = PSO(demo_func1, dim=20, pop=50, max_iter=100, lb=min_list[0], ub=max_list[0], w=0.8, c1=2, c2=2)
    pso.run()
    print('best_x is ', pso.gbest_x, 'best_y is', pso.gbest_y)
    plt.plot(pso.gbest_y_hist)
    plt.show()
    return pso.gbest_x
```

10.2 结果

问题一：筛选出的 20 个主要变量

主要变量	重要性
MDEC-23	0.194832414
maxHsOH	0.039845436
minsssN	0.03596156
C1SP2	0.034636028
maxssO	0.031497692
BCUTc-1l	0.01928962
minHBint5	0.015053126
VC-5	0.011328726
ATSc3	0.008373052
BCUTc-1h	0.006859063
MDEC-33	0.006858184
SHBint10	0.006847212
XLogP	0.006299382
mindssC	0.006250166
ETA_Shape_Y	0.005914768
BCUTp-1h	0.005407021
minHBa	0.005201538
ATSc2	0.005135356
VCH-5	0.004697659
WTPT-4	0.004469484

问题二：ERα_activity.xlsx 中 test 子表所有 50 种化合物的 pIC50 预测值

	SMILES	IC50_nM	pIC50
2	COc1cc(OC)cc(\C=C\c2ccc(OS(=O)(=O)[C@@H]3C[C@@H]4O[C@H]3C(=	22026.64	6.989311
3	OC(=O)\C=C\c1ccc(cc1)C2=C(CCOc3ccccc23)c4ccc(O)cc4	25881.84	7.081222
4	COc1ccc2C(=C(CCOc2c1)c3ccc(O)cc3)c4ccc(\C=C\C(=O)O)cc4	25881.84	7.082384
5	OC(=O)\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	25881.84	7.083121
6	OC(=O)\C=C\c1ccc(cc1)C2=C(CCSc3cc(F)ccc23)c4ccc(O)cc4	20344.74	7.044297
7	CC(=O)\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	21557.64	7.128607
8	Oc1ccc(cc1)C2=C(c3ccc(\C=C\c4ccccc4)cc3)c5ccc(F)cc5OCC2	21557.64	7.083542
9	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)c4ccccc4)cc3)c5ccc(F)cc5OCC2	21557.64	7.124861
10	OC(=O)\C=C\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	25881.84	7.081222
11	CCN(CC)C(=O)\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	9562.011	7.616419
12	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)N4CCCC4)cc3)c5ccc(F)cc5OCC2	9562.011	7.663398
13	CCN(CC)CCNC(=O)\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	9562.011	7.582197
14	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)N4CCNCC4)cc3)c5ccc(F)cc5OCC2	168379	8.023134
15	CN1CCN(CC1)C(=O)\C=C\c2ccc(cc2)C3=C(CCOc4cc(F)ccc34)c5ccc(O)cc5	9562.011	7.611894
16	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)N4CCN(Cc5ccccc5)CC4)cc3)c6ccc(F)c(	9562.011	7.62453
17	Cc1ccc(cc1)N2CCN(CC2)C(=O)\C=C\c3ccc(cc3)C4=C(CCOc5cc(F)ccc45)c6(	9562.011	7.64038
18	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)Nc4ccccc4)cc3)c5ccc(F)cc5OCC2	21375.84	7.12165
19	OC(=O)COc1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	22264.64	7.080175
20	Oc1ccc(cc1)C2=C(c3ccc(C=O)cc3)c4ccc(F)cc4OCC2	22735.34	7.10283
21	CCC(=C(c1ccc(O)cc1)c2ccc(\C=C\C(=O)O)cc2)c3ccccc3	25881.84	6.739807
22	OC(=O)CCCOc1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	174837.2	7.483183
23	COc1ccc2C(=C(CCOc2c1)c3ccc(O)cc3)c4ccc(OCC(=O)O)cc4	26951.84	7.088346
24	CCCC(CCC)(c1ccc(O)c(C)c1)c2cccs2CCCC(CCC)(c1ccc(O)c(C)c1)c2cccs2	15625.24	7.217232
25	CCCC(CCC)(c1ccc(O)c(C)c1)c2cc(C)cs2	15625.24	7.222569
26	CCCC(CCC)(c1ccc(O)c(C)c1)c2ccc([nH]2)C(=O)OCC	20724.38	7.010247
27	CCCC(CCC)(c1ccc(O)c(C)c1)c2ccc(C(=O)OCC)n2C	20724.38	7.036739
28	CCCC(CCC)(c1ccc(O)c(C)c1)c2c[nH]c3cc(OCc4ccccc4)ccc23	21665.34	7.071321
29	Oc1ccc(cc1)c2nc(Cl)c(c(Oc3ccc(OCCN4CCCC4)cc3)n2)c5ccccc5	9514.311	7.591994
30	CN(C)CCOc1ccc(Nc2nc(nc(Cl)c2c3ccccc3)c4ccc(O)cc4)cc1	9514.311	7.980461
31	COc1ccc(cc1)c2nc(Cl)c(c(Oc3ccc(OCCN4CCCC4)cc3)n2)c5ccccc5	9530.911	7.593752
32	COc1ccc(cc1)c2nc(Cl)cc(Oc3ccc(OCCN4CCCC4)cc3)n2	10556.91	7.607501
33	CCN(CC)CCOc1ccc(Oc2cc(Cl)nc(n2)c3ccc(OC)cc3)cc1	9530.911	7.576658
34	COc1ccc(cc1)c2nc(Cl)cc(Oc3ccc(OCCN(C)C)cc3)n2	3165.4	7.607501
35	CCN(CC)CCOc1ccc(Nc2cc(Cl)nc(n2)c3ccc(OC)cc3)cc1	9530.911	8.052527
36	COc1ccc(cc1)c2nc(Cl)cc(Nc3ccc(NC(=O)CN4CCCC4)cc3)n2	3248.7	7.886212
37	CO[C@H]1C[C@H](C)CC2=C(NC\C=C\CCC[C@@H]3C[C@H]4[C@@H]5(	9570.011	7.035479
38	CO[C@H]1C[C@H](C)CC2=C(NCC#CCC[C@@H]3C[C@H]4[C@@H]5CCc	9570.011	6.969161
39	C[C@]12CC[C@H]3[C@@H](CCc4cc(O)ccc34)[C@@H]1CC[C@@]2(O)C#(	20724.38	7.006254
40	CC(C)C[C@H](NC(=O)[C@](C)(CCCC=C)NC(=O)[C@H](CCC(=O)N)NC(=	16573.13	6.384947
41	CC(C)C[C@H](NC(=O)[C@]1(C)CCC\C=C/CCC[C@](C)(NC(=O)[C@H](CC	16573.13	6.449697
42	CC(C)C[C@H](NC(=O)[C@]1(C)CCC\C=C/C[C@@H](C)C[C@](C)(NC(=O)	16573.13	6.476582
43	CC(C)C[C@H](NC(=O)[C@]1(C)C[C@H](C)C\C=C/CCC[C@](C)(NC(=O)[C	16573.13	6.476582
44	CC(C)C[C@H](NC(=O)[C@]1(C)CCC\C=C/CCC[C@](C)(NC(=O)[C@H](CC	16573.13	6.384947
45	CC[C@H](C)[C@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](Cc1cnc[nH]1)N	16573.13	6.413789
46	CC(C)C[C@H](NC(=O)[C@]1(C)CCC\C=C/C[C@H](C)C[C@](C)(NC(=O)[C	16573.13	6.476582
47	CC\C(=C(/c1ccc(O)cc1)\c2ccc(OCCN(C)C)cc2)\c3ccc(cc3)C(=O)NO	14148.97	7.547825
48	CC\C(=C(/c1ccc(O)cc1)\c2ccc(OCCN(C)C)cc2)\c3ccc(CCCCC(=O)NO)cc3	14149.34	7.561955
49	CC\C(=C(/c1ccc(O)cc1)\c2ccc(OCCN(C)C)cc2)\c3ccc(CCC(=O)NO)cc3	14149.34	7.547825
50	CC\C(=C(/c1ccc(O)cc1)\c2ccc(OCCN(C)C)cc2)\c3ccc(OCCCCC(=O)NO)cc3	14382.83	7.558245
51	CCC(=C(c1ccc(O)cc1)c2ccc(OCCN3CCCC3)cc2)Cc4ccccc4	10225.28	7.693891

问题三：ADMET.xlsx 中 test 子表所有 50 种化合物的 ADMET 特性预测

1	SMILES	Caco-2	CYP3A4	hERG	HOB	MN
2	COc1cc(OC)cc(\C=C\c2ccc(OS(=O)(=O)[C@@H]3C[C@@H]4O[C@H]3C(=	0	1	0	0	1
3	OC(=O)\C=C\c1ccc(cc1)C2=C(CCOc3ccccc23)c4ccc(O)cc4	0	1	1	0	1
4	COc1ccc2C(=C(CCOc2c1)c3ccc(O)cc3)c4ccc(\C=C\C(=O)O)cc4	0	1	1	0	1
5	OC(=O)\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	0	1	1	0	1
6	OC(=O)\C=C\c1ccc(cc1)C2=C(CCSc3cc(F)ccc23)c4ccc(O)cc4	0	1	1	0	1
7	CC(=O)\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	0	1	1	0	1
8	Oc1ccc(cc1)C2=C(c3ccc(\C=C\c4ccccc4)cc3)c5ccc(F)cc5OCC2	0	1	1	0	1
9	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)c4ccccc4)cc3)c5ccc(F)cc5OCC2	0	1	0	0	1
10	OC(=O)\C=C\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	0	1	1	0	1
11	CCN(CC)C(=O)\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	0	1	0	0	1
12	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)N4CCCC4)cc3)c5ccc(F)cc5OCC2	0	1	0	0	1
13	CCN(CC)CCNC(=O)\C=C\c1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	0	1	0	0	1
14	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)N4CCNCC4)cc3)c5ccc(F)cc5OCC2	0	1	0	0	1
15	CN1CCN(CC1)C(=O)\C=C\c2ccc(cc2)C3=C(CCOc4cc(F)ccc34)c5ccc(O)cc5	0	1	0	0	1
16	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)N4CCN(Cc5ccccc5)CC4)cc3)c6ccc(F)cc	0	1	0	0	1
17	Cc1ccc(cc1)N2CCN(CC2)C(=O)\C=C\c3ccc(cc3)C4=C(CCOc5cc(F)ccc45)c6c	0	1	0	0	1
18	Oc1ccc(cc1)C2=C(c3ccc(\C=C\C(=O)Nc4ccccc4)cc3)c5ccc(F)cc5OCC2	0	1	1	0	1
19	OC(=O)COc1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	0	1	1	0	1
20	Oc1ccc(cc1)C2=C(c3ccc(C=O)cc3)c4ccc(F)cc4OCC2	0	1	1	0	1
21	CCC(=C(c1ccc(O)cc1)c2ccc(\C=C\C(=O)O)cc2)c3ccccc3	0	1	1	0	1
22	OC(=O)CCCOc1ccc(cc1)C2=C(CCOc3cc(F)ccc23)c4ccc(O)cc4	0	1	1	0	1
23	COc1ccc2C(=C(CCOc2c1)c3ccc(O)cc3)c4ccc(OCC(=O)O)cc4	0	1	1	0	1
24	CCCC(CCC)(c1ccc(O)c(C)c1)c2cccs2CCCC(CCC)(c1ccc(O)c(C)c1)c2cccs2	0	1	1	1	1
25	CCCC(CCC)(c1ccc(O)c(C)c1)c2cc(C)cs2	0	1	1	1	1
26	CCCC(CCC)(c1ccc(O)c(C)c1)c2ccc([nH]2)C(=O)OCC	0	1	0	0	1
27	CCCC(CCC)(c1ccc(O)c(C)c1)c2ccc(C(=O)OCC)n2C	0	1	1	1	1
28	CCCC(CCC)(c1ccc(O)c(C)c1)c2c[nH]c3cc(OCc4ccccc4)ccc23	0	1	1	1	1
29	Oc1ccc(cc1)c2nc(Cl)c(c(Oc3ccc(OCCN4CCCC4)cc3)n2)c5ccccc5	0	1	1	0	1
30	CN(C)CCOc1ccc(Nc2nc(nc(Cl)c2c3ccccc3)c4ccc(O)cc4)cc1	0	1	1	1	1
31	COc1ccc(cc1)c2nc(Cl)c(c(Oc3ccc(OCCN4CCCC4)cc3)n2)c5ccccc5	0	1	1	0	1
32	COc1ccc(cc1)c2nc(Cl)cc(Oc3ccc(OCCN4CCCC4)cc3)n2	0	1	1	0	1
33	CCN(CC)CCOc1ccc(Oc2cc(Cl)nc(n2)c3ccc(OC)cc3)cc1	0	1	1	0	1
34	COc1ccc(cc1)c2nc(Cl)cc(Oc3ccc(OCCN(C)C)cc3)n2	0	1	1	0	1
35	CCN(CC)CCOc1ccc(Nc2cc(Cl)nc(n2)c3ccc(OC)cc3)cc1	0	1	1	0	1
36	COc1ccc(cc1)c2nc(Cl)cc(Nc3ccc(NC(=O)CN4CCCC4)cc3)n2	0	1	0	0	1
37	CO[C@H]1C[C@H](C)CC2=C(NC\C=C\CCC[C@@H]3C[C@H]4[C@@H]5(	0	1	1	0	1
38	CO[C@H]1C[C@H](C)CC2=C(NCC#CCC[C@@H]3C[C@H]4[C@@H]5CCc	0	1	1	0	1
39	C[C@]12CC[C@H]3[C@@H](CCc4cc(O)ccc34)[C@@H]1CC[C@@]2(O)C#(	0	1	1	0	1
40	CC(C)C[C@H](NC(=O)[C@](C)(CCCC=C)NC(=O)[C@H](CCC(=O)N)NC(=	0	1	1	0	1
41	CC(C)C[C@H](NC(=O)[C@]1(C)CCC\C=C/CCC[C@](C)(NC(=O)[C@H](CC	0	1	1	0	1
42	CC(C)C[C@H](NC(=O)[C@]1(C)CCC\C=C/C[C@@H](C)C[C@](C)(NC(=O)	0	1	1	0	1
43	CC(C)C[C@H](NC(=O)[C@]1(C)C[C@H](C)C\C=C/CCC[C@](C)(NC(=O)[C	0	1	1	0	1
44	CC(C)C[C@H](NC(=O)[C@]1(C)CCC\C=C/CCC[C@](C)(NC(=O)[C@H](CC	0	1	1	0	1
45	CC[C@H](C)[C@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](Cc1cnc[nH]1)N	0	1	1	0	1
46	CC(C)C[C@H](NC(=O)[C@]1(C)CCC\C=C/C[C@H](C)C[C@](C)(NC(=O)[C	0	1	1	0	1
47	CC\C(=C(/c1ccc(O)cc1)\c2ccc(OCCN(C)C)cc2)\c3ccc(cc3)C(=O)NO	0	1	1	1	1
48	CC\C(=C(/c1ccc(O)cc1)\c2ccc(OCCN(C)C)cc2)\c3ccc(CCCCC(=O)NO)cc3	0	1	1	1	1
49	CC\C(=C(/c1ccc(O)cc1)\c2ccc(OCCN(C)C)cc2)\c3ccc(CCC(=O)NO)cc3	0	1	1	1	1
50	CC\C(=C(/c1ccc(O)cc1)\c2ccc(OCCN(C)C)cc2)\c3ccc(OCCCCC(=O)NO)cc3	0	1	0	0	1
51	CCC(=C(c1ccc(O)cc1)c2ccc(OCCN3CCCC3)cc2)Cc4ccccc4	0	1	1	1	1