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		上海大学
参赛队号		**21102800176**
队员姓名	**1.**	顾东洲
	2.	俞寅生
	3.	刘骁

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：

乳腺癌是女性癌症高发性恶性肿瘤，近年来发病率和死亡率逐年上升，严重危害了女性健康。如何使用数学模型辅助专家高效研发抗乳腺癌药物具有重要意义。本文通过构建化合物的定量结构-活性关系(QSAR)模型来筛选潜在活性化合物，使其不仅具有较好的生物活性，同时在人体内具备良好的药代动力学性质和安全性。具体做法如下：

针对问题 1，为了筛选出 1974 个化合物的 729 个分子描述符中对生物活性影响最显著的前 20 个变量，建立了**分步集成筛选模型**，首先对所有分子描述符**利用方差信息**分别对整型变量和浮点型变量进行**无关特征过滤**，剩余 341 维特征；接着，使用**相关性系数分步**对描述符类内与类间的**冗余特征过滤，**得到 137 维特征；然后采用**集成式特征筛选模型**选出重要性排名前 20 名的分子描述符。具体来说，集成式筛选模型分别选用了过滤式特征筛选中的 **Spearman 系数法，距离相关系数法**和嵌入式特征筛选中的**随机森林法**和**弹性网络法**，将四种方法给出的特征重要性集成并排序，选择出排名前 20 名的分子描述符；最后，对选出的显著变量使用 Spearman 系数进行**独立性验证**，并将集成特征筛选法与各个特征筛选法得出主要变量送入主流机器学习预测方法中对比回归性能，验证了筛选出的 20 个变量的**代表性**，证明了分步集成特征筛选模型的有效性与优势。

针对问题 2，题目要求在选择不超过 20 个分子描述符变量的前提下，构建化合物的生物活性定量预测模型。此题为**问题一的延申**，我们选取了问题一中得出的 20 个特征作为化合物生物活性定量预测模型的输入，pIC_{50} 生物活性值作为模型的预测目标。首先，我们**观察**了题目给出的训练集与测试集的**原始分布**；接着构建了**基于 Stacking 的集成学习预测模型**和**基于深度森林（Deep Forest）的预测模型**，并使用 TPE 方法对模型参数调优；最后，通过详细的**对比实验**验证了 Deep Forest 模型不仅更高的预测性能，而且训练时间更短，最终选取深度森林预测模型作为化合物的生物活性定量预测模型，并对测试集中 50 种化合物的 IC_{50} 和 pIC_{50} 值进行预测。

针对问题 3, 根据化合物的 729 个分子描述符，对化合物 ADMET 的 5 个性质(Caco-2、CYP3A4、hERG、HOB、MN)分别进行**二分类预测**。本题没有限制特征的数量，基于对模型简单、通用、特征提取能力强大的考量，构建了**基于原型网络的小样本学习分类模型**。首先，对样本中的 ADMET 性质进行数据分析，发现了化合物 ADMET 性质中的 CYP3A4, HOB, MN 存在明显的**类别不平衡**现象，为此，采用 **Borderline-SMOTE 过采样**算法进行处理；接着，考虑到神经网络强大的非线性特征提取能力，将其作为原型网络的嵌入函数，自动提取特征并利用反向传播算法拟合嵌入函数的参数空间；本问选取了第一问中剔除了类内类间冗余特征的 137 个变量作为原型网络的输入特征，对模型训练并可视化优化过程

与分类结果，并对测试集中 50 个化合物的 ADMET 性质进行预测。 最后，使用 **MIV 算法**得出本题建立的原型网络对 137 个自变量的特征重要性排序，并分别计算出影响每个原型网络分类器的前 20 个显著特征。

针对问题 4，为了寻找化合物的哪些分子描述符对抑制 ERα 具有更好的生物活性且使得化合物具有更好的 ADMET 性质（给定的五个 ADMET 性质中，至少三个性质较好），建立了**基于初始值优化的粒子群算法**的**多目标混合整数规划模型**，将化合物的pIC_{50}值和 ADMET 性质作为组合优化目标，一共选取了 137 个分子描述符作为模型的**决策变量**，由于决策变量的取值范围难以通过专家经验确定，所以选取了 1974 个样本中分子描述符的最大取值和最小取值作为决策变量的取值范围，并且对 137 个分子描述符中整型变量进行**整型约束**。由于搜索空间较大，所以对粒子群算法进行了优化，对初始值的选取融合了**专家经验**，最后得出 23 个分子描述符对化合物的pIC_{50}值影响较大，并给出了它们的取值或取值范围，同时证明了改进粒子群算法的有效性。

最后我们将问题 4 优化模型得出的 23 个分子描述符、问题 2 预测模型采用的 20 个分子描述符和问题 3 中原型网络分类模型得出的 61 个分子描述符之间进行了**重叠性分析**并绘制了**维恩图**，发现有部分分子描述符之间存在交集，且 **MDEC-23、SaasC 和 C3SP2** 这三个分子描述符在四个问题中均出现，这直接证明了其对化合物性质影响较大，并侧面说明了问题 1，2，3，4 模型的有效性。

关键词：QSAR，分步集成筛选模型，深度森林（Deep Forest），原型网络，粒子群优化

目 录

图 录

表 录

一、问题重述

1.1 问题背景

乳腺癌是女性癌症高发性恶性肿瘤，其发病率和病死率逐年上升，已经成为世界以及中国女性发病率最高的恶性肿瘤。数据统计显示，2016 年中国新发乳腺癌病例数估算为 27.2 万，有 7.1 万患者死于乳腺癌[1]。

乳腺癌的发病因素复杂，是多种因素综合作用的结果，至今还无法用已知的单因素或多因素模型来全面解释乳腺癌的发生和发展机制与过程。研究发现，雌激素受体 α 亚型（Estrogen receptors alpha, ERα）在不超过 10%的正常乳腺上皮细胞中表达，但大约在 50%-80%的乳腺肿瘤细胞中表达；而对 ERα 基因缺失小鼠的实验结果表明，ERα 确实在乳腺发育过程中发挥着重要作用。因此，ERα 被认为是治疗乳腺癌的重要靶标，在分子水平对其通路靶点设计药物, 通过药物与受体或调节分子结合，下调受体表达或者活化下游基因, 使得癌细胞凋亡或者抑制其生长。目前，在临床治疗中能够拮抗 ERα 活性的经典药物有莫昔芬、雷诺昔芬等。

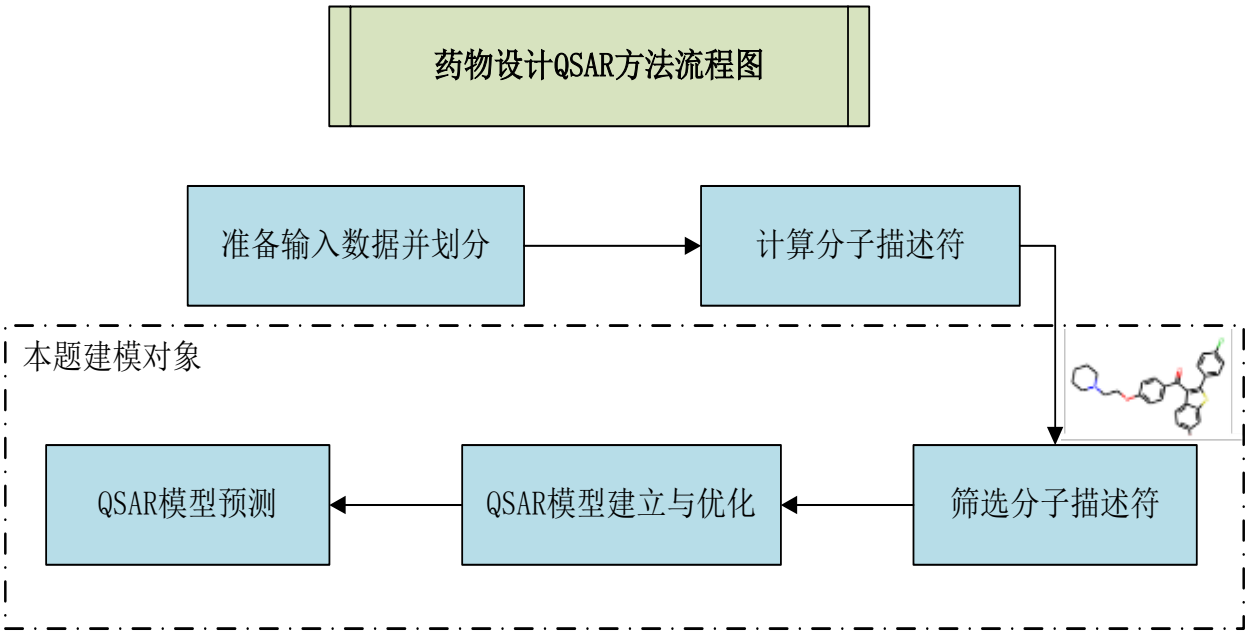


图 1 药物设计 QSAR 方法流程图

在药物研发过程中，传统的高通量筛选（High throughput screening，HTS）耗时且昂贵，目前应用最为广泛的是构建定量构效关系（Quantitative Structure-Activity Relationship, QSAR）模型来筛选潜在活性化合物，即通过数学方法建立化合物的分子描述符与其生物活性/毒性之间的线性或非线性关系模型，于分子水平阐明结构与生物学及物理化学特性之间的关系。具体做法如图 1 所示，以本题乳腺癌为例：1）首先，针对与乳腺癌相关的靶标 ERα，收集一系列作用于该靶标的化合物及其生物活性数据，并划分训练集测试集；2）接着，计算化合物的一系列分子描述符如拓扑指数、几何描述符等作为化合物的自变量特征；3）为了去除不相关的描述符，通常需要对其合理筛选，选择影响比重较大的若干分

子描述符，可以有效地排除噪音数据干扰，从而提升模型的构建速度和质量；4）然后采用机器学习或统计学方法，建立分子描述符与研究性质的数学关系。5）最后，利用训练数据集和优化算法构建 QSAR 模型后，使用测试数据集对模型进行测试，产生对药物研发有价值的药物特性预测结果。本题的研究对象为后三步即：筛选分子描述符、QSAR 模型建立与优化及 QSAR 模型预测。

1.2 问题的提出

针对乳腺癌治疗靶标 Erα，根据采集的 1974 个化合物信息（每个样本都有 729 个分子描述符变量，1 个生物活性数据，5 个 ADMET 性质数据），使用数据挖掘技术建立化合物生物活性的定量预测模型和 ADMET 性质的分类预测模型，从而为同时优化 ERα 拮抗剂的生物活性和 ADMET 性质提供预测服务。现根据以上背景以及所提供数据完成以下任务：

问题 1：特征选择与重要性排序

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，给出前 20 个对生物活性最具有显著影响的分子描述符，并给出具体筛选过程及合理性验证。

问题 2：构建定量预测模型

结合问题 1 的特征选择结果，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，并给出文件“ERα_activity.xlsx”的 test 表中的 50 个化合物的 IC_{50} 值和 pIC_{50} 值的预测结果，并将结果分别填入表中。

问题 3：分别构建 5 个二分类模型

提供 729 个分子描述符变量，针对 “ADMET.xlsx”文件中提供的 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，并给出具体建模过程，然后使用构建的模型对测试集中的 50 个化合物进行分类预测，并填入对应表中。

问题 4：寻找变量的最优范围

在使化合物对抑制 ERα 具有更好的生物活性，同时具有更好的 ADMET 性质（至少三个性质较好）的**目标和约束下**，寻找并阐述化合物的分子描述符的最优取值范围。

二、总体技术路线图

本文技术路线如下：

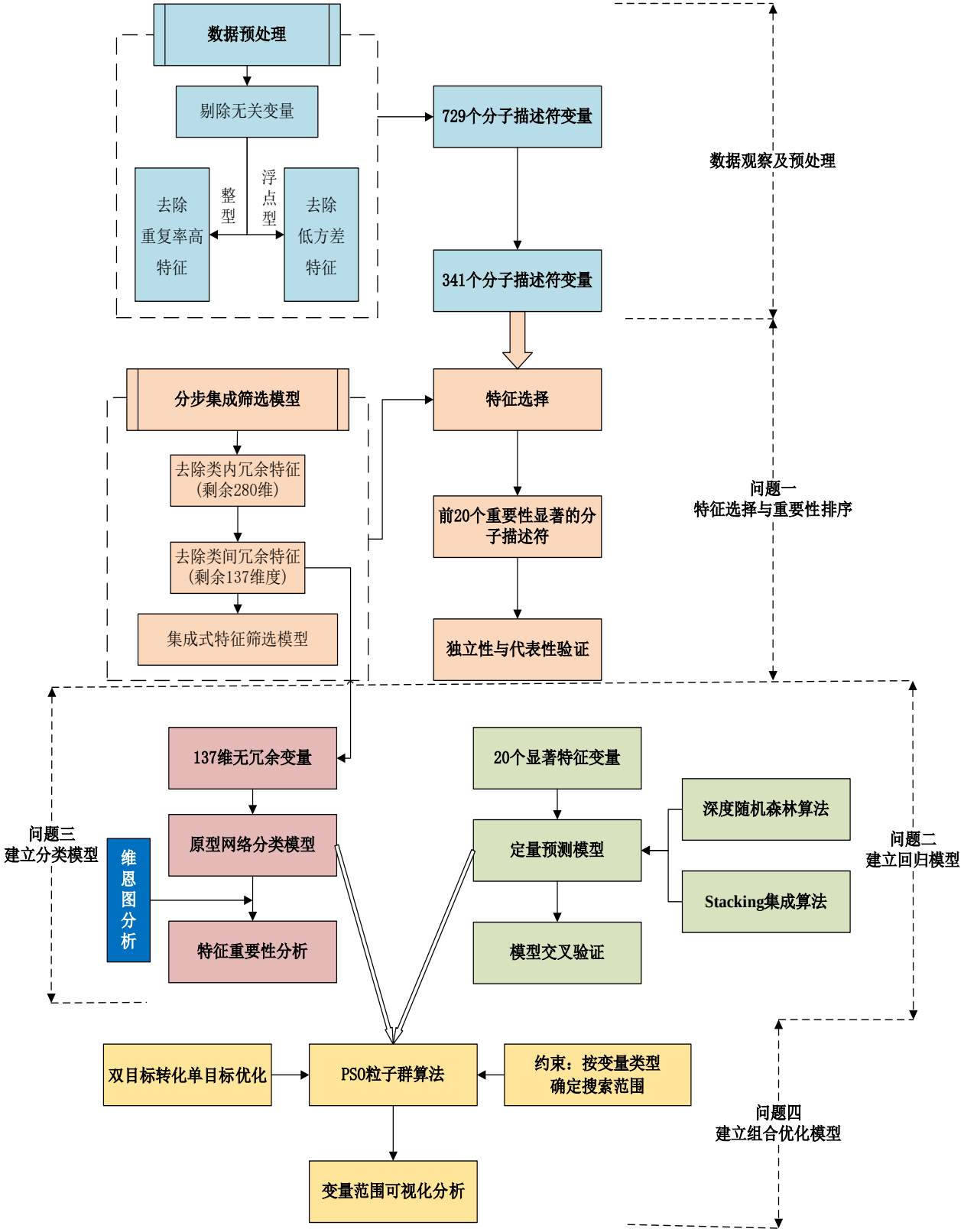

数据预处理
剔除无关变量
去除
重复率高
特征
整型
浮点型
去除
低方差
特征
729个分子描述符变量
341个分子描述符变量
数据观察及预处理
分步集成筛选模型
去除类内冗余特征
(剩余280维)
去除类间冗余特征
(剩余137维度)
集成式特征筛选模型
特征选择
前20个重要性显著的分子描述符
独立性与代表性验证
问题一
特征选择与重要性排序
137维无冗余变量
原型网络分类模型
维恩图分析
特征重要性分析
问题三
建立分类模型
20个显著特征变量
定量预测模型
深度随机森林算法
Stacking集成算法
模型交叉验证
问题二
建立回归模型
双目标转化单目标优化
PSO粒子群算法
约束：按变量类型
确定搜索范围
变量范围可视化分析
问题四
建立组合优化模型

图 2 全文技术路线图

三、基本假设与符号说明

3.1 基本假设

假设 1：本题中所提供的 1974 个化合物的 729 个分子描述符均是准确的，不存在异常值；

假设 2：本题中所提供的 1974 个化合物的活性和 ADMET 性质都是准确的；

假设 3：除了题目提供的 729 个分子描述符之外，不存在其他因素影响化合物活性和 ADMET 性质；

假设 4：题目提供的训练数据和测试数据的原始分布一致；

假设 5：问题 4 中，启发式优化算法使用的预测模型和分类模型结果准确。

3.2 符号说明

符号表示	含义说明
r	皮尔森相关系数
ρ	Spearman 相关系数
τ_B	肯德尔系数
$R^2(x,y)$	距离相关系数
n_k	决策树中，计算某一个节点 k 的重要性
c_k	原型
$p_\phi(y=k\|x)$	原型的概率分布
IC_{50}	化合物对 $ER\alpha$ 的生物活性值
pIC_{50}	化合物对 $ER\alpha$ 的生物活性值的负对数
v_{id}	粒子 i 飞行速度矢量的第 d 维分量
x_{td}	粒子 i 位置矢量的第 d 维分量
c_1、c_2	粒子群的学习因子
r_1、r_2	粒子群的随机数

ω	粒子群的惯性权重
z_1	pIC_{50}的优化目标
z_2	$ADMET$性质的优化目标
$\mathbb{Z}_{admet}$	$ADMET$性质的集合
X	经过冗余筛选后的分子描述符集合
X_z	经过冗余筛选后的分子描述符中整型变量集合
X_T	重要性排名前 20 个分子描述符排序
$[lower_i, upper_i]$	第i个分子描述符的取值范围
$f(\bullet)$	pIC_{50}定量预测模型的非线性函数描述
$h_i(\bullet)$	第i个$ADMET$性质分类模型的非线性函数描述

四、数据观察及预处理

本题提供了大量的数据信息，尤其是分子描述符更是高达 729 个，分子描述符本身是面对生物化学领域的特征，需要首先分析数据的统计特征，并通过可视化来指导下面的建模。其中包括数据集的可视化以及相应的统计信息，从而对数据有全局的印象。同时，针对可视化的结果指导数据预处理，包括对变化小的离散变量和连续变量。另外，由于各个基本信息的取值范围变化幅度不一，直接放入模型中会导致效果不佳，所以 还需要对数据做归一化处理，消除量纲影响，以便后期喂入模型。

4.1 数据宏观统计与可视化

首先，我们对 1974 个化合物的 729 个分子描述符进行宏观统计。通过遍历统计，发现变量没有缺失值，并且只有整型变量和浮点型变量，变量的类型分布图如下，离散的变量略多于连续变量，且没有其他复杂变量类型。

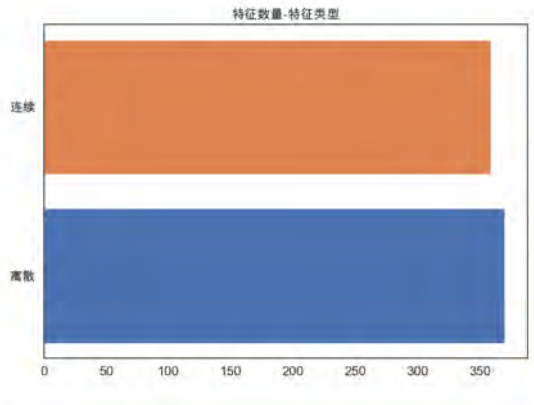


图 3 变量类型分布图

针对离散的整型变量，我们对共计 370 个变量进行统计，并绘制出直方图，希望可以对变量的原始分布有整体的印象，这里随机选择 16 个变量便于展示，见图 3 所示。可以发现，很多分子描述符变量的取值都为 0,1 数值，代表有无该分子描述符，还有许多分子描述符只有单个值，这些含有太多常值的描述符对每个化合物都一样，这样就对体系建模没有区别性贡献，在预处理操作中应删除。

针对连续的浮点型变量，我们对共计 359 个变量进行统计，并绘制每个变量的分布图，如图 4 所示。可以发现，Kier3、mindS、minddssS 等描述符大多数值都集中在某一值附近，存在低变化性的特征，这类标准偏差极小的变量也应在预处理操作中删除。

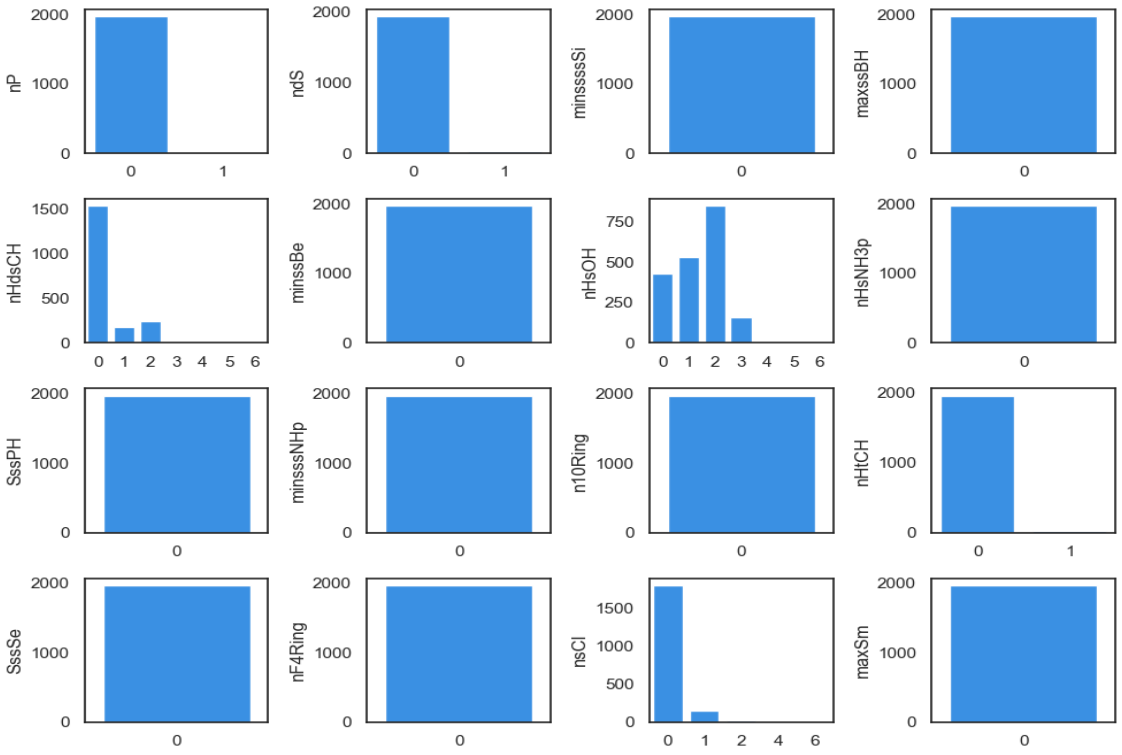


图 4 整型变量直方统计图

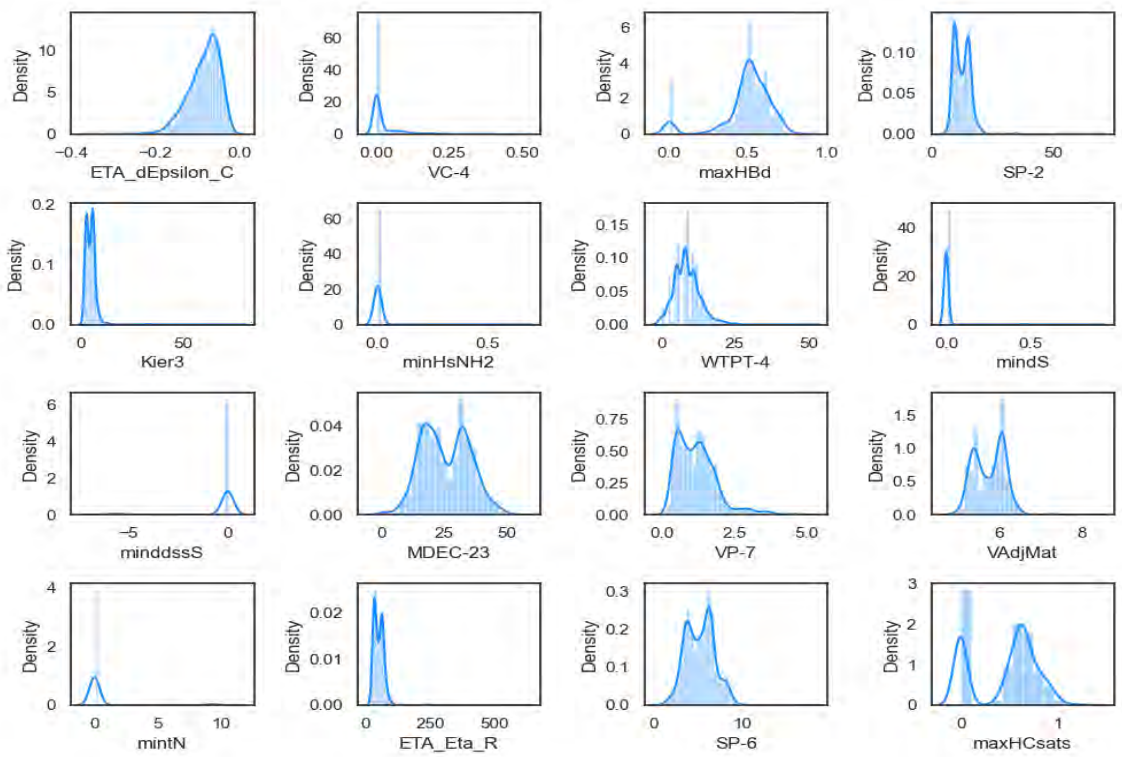


图 5 浮点型变量分布图

除此之外，题目还提供了这 729 个分子描述符的描述信息，表中给出了 53 类分子描述符，但通过对比发现，表中 53 类描述符包含的分子描述符有许多并不存在于 729 个变量，经过筛选，重新得到了 47 个类别的描述符数量，具体信息见表 1，可以发现所有分子描述符都为 2D 描述符（由 2D 分子图形或者结构片断计算得来的，如拓扑指数，2D 分子指纹，连接表，图，结构描述符等）。每个类别的分子描述符数量成长尾分布，见图 5 所示，其中有 488 个分子描述符类别为 Atom type electrotopological state（原子型电拓扑态），23 个类别中只有一个分子描述符。

表 1 分子描述符的类别统计

描述符类别	数量	类别	描述符变量
Atom type electrotopological state	488	2D	NHBd…
Extended topochemical atom	43	2D	ETA_Alpha…
Ring count	23	2D	NRing…
Molecular distance edge	19	2D	MDEC-11…
Chi path	16	2D	SP-0…
Atom count	14	2D	NAtom…
Chi chain	10	2D	SCH-3…
Bond count	10	2D	NBond…
Carbon types	9	2D	C1SP1…
Chi cluster	8	2D	SC-3

…	…		…
HybridizationRatioDescriptor	1	2D	HybRatio

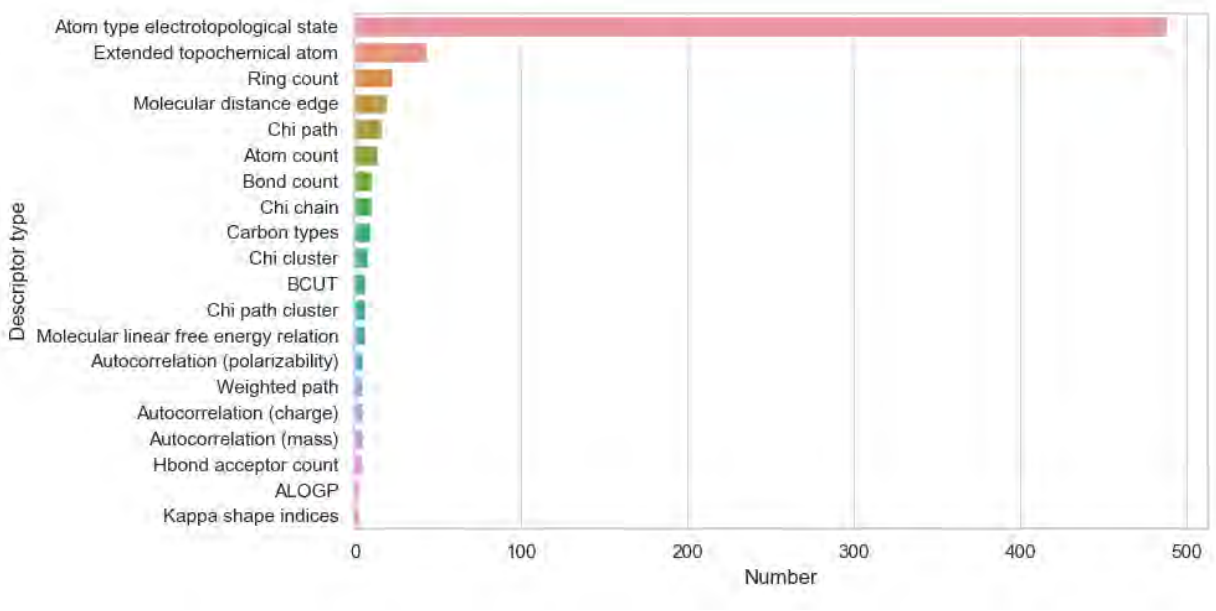


图 6 描述符类别与数量直方图

分析出同一类别内的分子描述符是否存在高相关性，可以为之后的建模做准备。因此，我们使用皮尔逊系数对同一类别下的分子描述符进行了数据观察，以此来发现类别内的变量相关性 缩略图见图 6 所示，可以发现，许多描述符类别下的变量具有很高的相关性，对于这样的高耦合的变量特征，应在后面的建模中考虑过滤筛选。

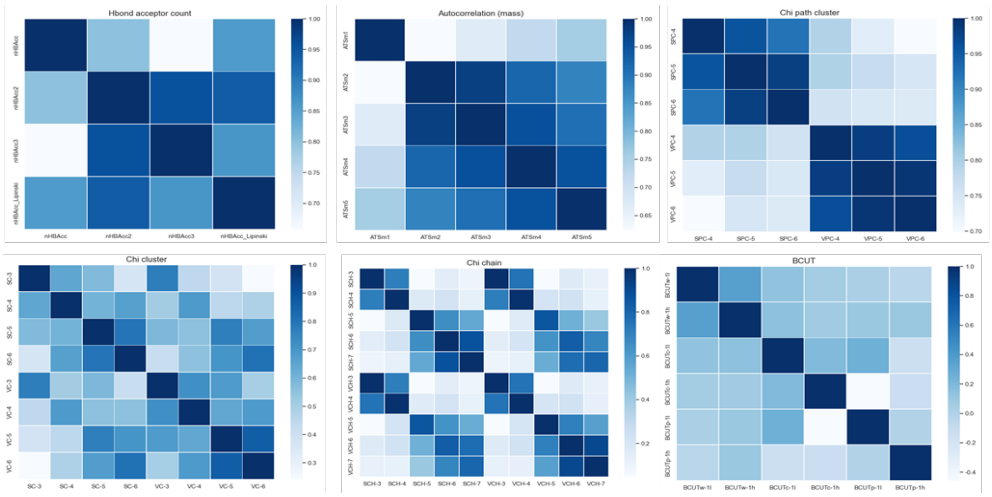


图 7 不同描述符类别下的变量相关性

4.2 数据预处理

数据预处理是数据挖掘中的重要步骤，不仅能够提高数据的质量，也能让数据更好地适应建立的模型，因此，针对 4.1 章节的数据宏观观察与可视化观察，我们进行了如下的数据处理：

步骤 1: 过滤高重复率的整型变量

如果该整型变量的 90%样本的描述符值是相同的，则说明该描述符对每个化合物都一样，属于无关特征，可以直接删除这个特征，公式如下，对于 *threshlod* 阈值的选择，我们从 0 至 1 每 0.1 取一点进行特征筛选数量对比，见下图所示，可以发现，重复率高、变化小的特征数量在 *threshlod = 0.9* 存在显著拐点， 因此我们将重复率大于 0.9 的特征视为无关特征（共计 362 个）并删除。

$$\max\left(\sum_{i=0}^{n} count(i)/n\right) > threshold \tag{1}$$

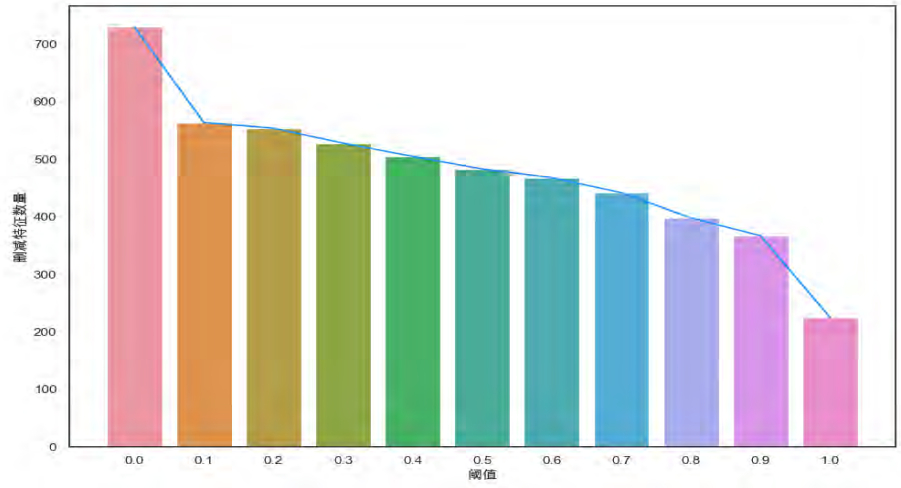


图 8 待删除特征数量与阈值的关系

步骤 2: 过滤方差极小的浮点型变量

根据对数据中 729 变量的分布统计与可视化，可以发现，存在变量严重集中在某一值附近的情况，对于这种方差小、变化很小的特征，需要根据阈值过滤，这里根据论文中的标准差阈值 0.05 进行筛选，共过滤了 21 个分子描述符变量。

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}} \tag{2}$$

4.3 结论

通过对数据的观察和可视化，对本题给出的数据有了更全面的认识，便于指导后续模型的建立。同时，根据可视化的结果推动了对数据的预处理，筛除了无关特征，剩余 341 个特征，大大减少了特征的维度，减少了后续变量选择的难度。

五、问题一的求解：分步集成筛选模型

5.1 问题分析

根据问题一要求，需要对 729 个分子描述符进行变量选择，选择出对生物活性 pIC_{50} 具有显著影响的前 20 个分子描述符（即变量）。观察数据，采用软件计算得到的分子描述符多达 729 个，而数据样本只有不到两千个，而且某些分子描述符具有密切相关性，这些冗余描述符可能会导致后续 QSAR 模型的过拟合。因此，为了模型优化，对于这样的高维小样本数据，我们通常需要对计算出的描述符进行预处理，筛选出重要且独立性好的特征[2]。基于此，题目中要求的前 20 个对生物活性最具有显著影响的分子描述符，我们认为是**不包含高度相关性的重复特征**，是**独立性较好**同时具有**显著重要性**的特征。

特征选择不同于特征提取方法，特征提取是一种基于变换的方法，通过坐标变换的方式将原始的高维数据映射到低维空间，而特征选择可以看作从初始特征空间搜索出一个最优特征子集的过程，其并不改变原始特征空间，只是选择一些分辨力好的特征，组成一个新的低维空间，可以保留原始特征空间大部分性质并去除不相关特征和冗余特征，在一定程度上将噪声数据对分类器性能的影响降到最低。

基于上述分析和特征选择方法，问题一的整体思路流程见下图所示。经过第 4 章的数据预处理，无关特征已经被过滤，还剩余 341 维有效变量。同时，通过描述符类别内的相关性分析可视化图 6，可以发现，类内特征存在很大的冗余，因此采用**分步走**的方法

1) 首先基于相关系数建立**冗余特征**的筛选模型；

2) 接着，分别使用线性、非线性以及嵌入式的特征筛选模型构建特征子集，并得到特征的重要性排序；

3) 然后，依据特定的规则建立**集成筛选模型**，以弥补不同方法间的不足。

4) 最后，对分步集成筛选模型进行显著变量的**独立性验证**和**代表性验证**，说明方法的合理性。

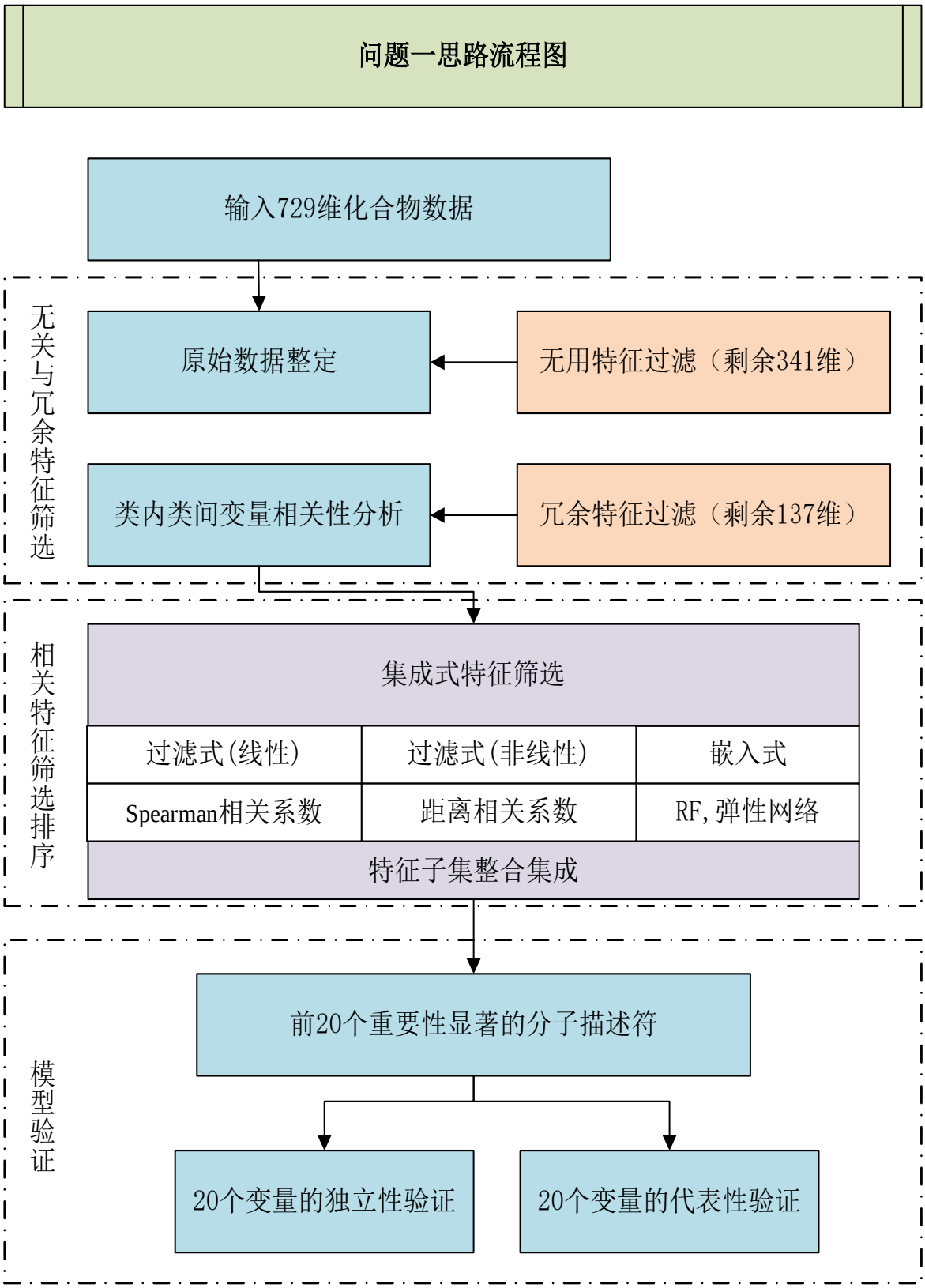

问题一思路流程图
输入729维化合物数据
无关与冗余特征筛选
原始数据整定
无用特征过滤（剩余341维）
类内类间变量相关性分析
冗余特征过滤（剩余137维）
相关特征筛选排序
集成式特征筛选
过滤式(线性)
过滤式(非线性)
嵌入式
Spearman相关系数
距离相关系数
RF,弹性网络
特征子集整合集成
模型验证
前20个重要性显著的分子描述符
20个变量的独立性验证
20个变量的代表性验证

图 9 问题一思路流程图

5.2 冗余特征过滤模型的建立

针对 47 类分子描述符的可视化结果，发现类内的分子描述符相关系数很高，具有高冗余性，因此，在本小节内，建立基于此的数学模型。

常用的相关性系数矩阵法主要包含 Pearson、Spearman、Kendall 三种典型算法，下面比较三种算法对本题的适用性。

皮尔逊(Pearson) 相关系数：又称为皮尔逊积矩相关系数[3]，是用于度量两个变量 X 和 Y 之间的相关性，其范围为-1 之 1 之间。一般用于分析两个变量的之间的关系，是一种线性相关系数，公式定义为两个变量之间的协方差和标准差的商：

$$r=\frac{\sum_{i=1}^{n}\left(X_i-\overline{X}\right)\left(Y_i-\overline{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i-\overline{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i-\overline{Y}\right)^2}} \tag{3}$$

Pearson 系数适用于：

(1) 两个变量之间是线性关系，都是连续数据;

(2) 两个变量的总体是正态分布，或接近正态的单峰分布;

(3) 两个变量的观测值是成对的，每对观测值之间相互独立。

斯皮尔曼(Spearman)相关系数: 又称斯皮尔曼秩相关系数，是秩相关系数的一种。“秩”，即秩序，可以理解为一种顺序或排序[4]，根据变量在数据内的位置进行计算，公式为：

$$\rho=1-\frac{6\sum d_i^2}{n(n^2-1)} \tag{4}$$

与皮尔逊相关系数相比，斯皮尔曼相关系数没有那些限制，不需要关心数据如何变化，符合什么样的分布，只需要关心每个变量对应数值的位置。如果两个变量的对应值，在各组内的排列顺位是相同或类似的，则具有显著的相关性。

肯德尔(Kendall)相关系数: 又称肯德尔秩相关系数，它也是一种秩相关系数，不过，它的目标对象是有序的类别变量，它可以度量两个有序变量之间单调关系强弱。肯德尔相关系数使用了“成对“这一概念来决定相关系数的强弱[5]，公式为：

$$\tau_B=\frac{n_c-n_d}{\sqrt{(n_0-n_1)(n_0-n_2)}} \tag{5}$$

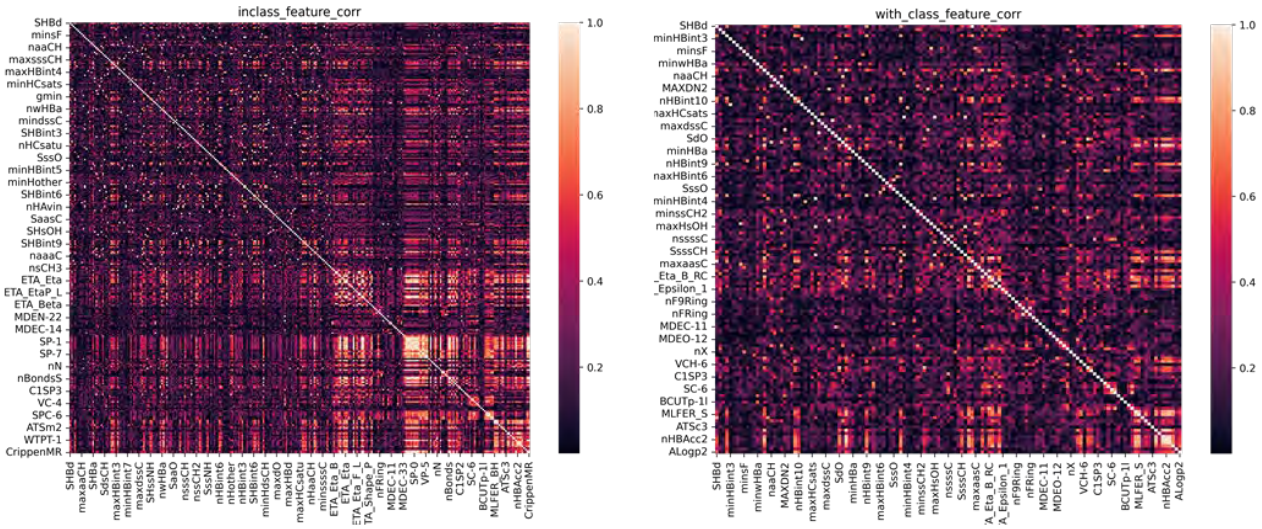
其中n_c表示 XY 中拥有一致性的元素对数；n_d表示 XY 中拥有不一致性的元素对数。与前两种方法相比，其更适用于有序的变量。

由于本题中的变量类型，还存在非连续型变量且不符合正态分布特性，因此选择斯皮尔曼系数进行冗余特征过滤数学建模。生物化合物结构数据中，一类的分子描述符往往表示出高度的相关性和冗余性，因此对于整理后的 47 类分子描述符分别进行类内的变量相关性检验，计算两两之间的 Spearman 系数，得到变量的相关性系数矩阵：

$$\text{Matrix}_{\rho} = \begin{bmatrix} \rho_{11}\rho_{12}\cdots\rho_{1D} \\ \rho_{21}\rho_{22}\cdots\rho_{2D} \\ \vdots \\ \rho_{D1}\rho_{D2}\cdots\rho_{DD} \end{bmatrix} \quad (6)$$

其中，ρ_{ij} 表示分子描述符变量 i 与变量 j 之间的 Spearma 相关系数值，D 表示当前的分子描述符数量，这里为预处理后的 341 维。然后，找出 ρ_{ij} 大于阈值 α 的位置，定义两个变量为强相关变量，按遍历顺序并将其中之一剔除，即可剔除类内强相关的冗余变量。

经过筛选后，最终剩余 280 维变量，筛选后的变量相关性如下图 9(a)所示，可以发现，右下角依然存在相关性较高的变量，因此，再剔除了类内冗余变量后，接着对类间的特征再逐步进行剔除，最后筛选出 137 维变量，其相关性热力图如图 9(b)所示，可以发现，剩余变量的相关性较低，独立性较高，满足冗余特征筛选要求。



(a) 280维　　(b) 137维

图 10 类内变量和类间变量筛选后的特征相关性热图

5.3 基于集成特征筛选模型的建立

随着特征选择方法不断出现，有一系列用于对数据进行特征选择的方法。大多数情况下，最常用的特征选择方法属于过滤法。另一方面，由于资源的大量计算消耗和过度拟合的高风险，包装器方法在文献中被大量避免。尽管嵌入式模型在数学建模的初期没有得到足够的重视，但近年来也出现了很多方法。值得注意的是，对最新文献的回顾显示无论集成方法在一些数据集上都取得了不错的效果[6]。

表 2 不同特征选择方法的优缺点

名称	优点	缺点
过滤式	优势：快速，可扩展的；独立于分类器；更好的泛化性能	分类精度一般，不能完全去冗余
包裹式	简单；与分类器模型交互；考虑特征间的相关性，比随机化方法计算量少	计算量大；只针对分类器的选择；有更高的过拟合风险
嵌入式	与分类器交互；比 wrapper 计算复杂性低，也不容易过拟合；比 filter 更高的性能精确度	只针对特定的分类器
集成式	不易过度拟合;对高维数据更加灵活和稳定	时间性能一般

由于本题的化合物数据为小样本高维数据，基于上表的考量，我们最终建立了集成特征筛选模型，以期望不同的模型能够互补劣势，其流程如图 10 所示。

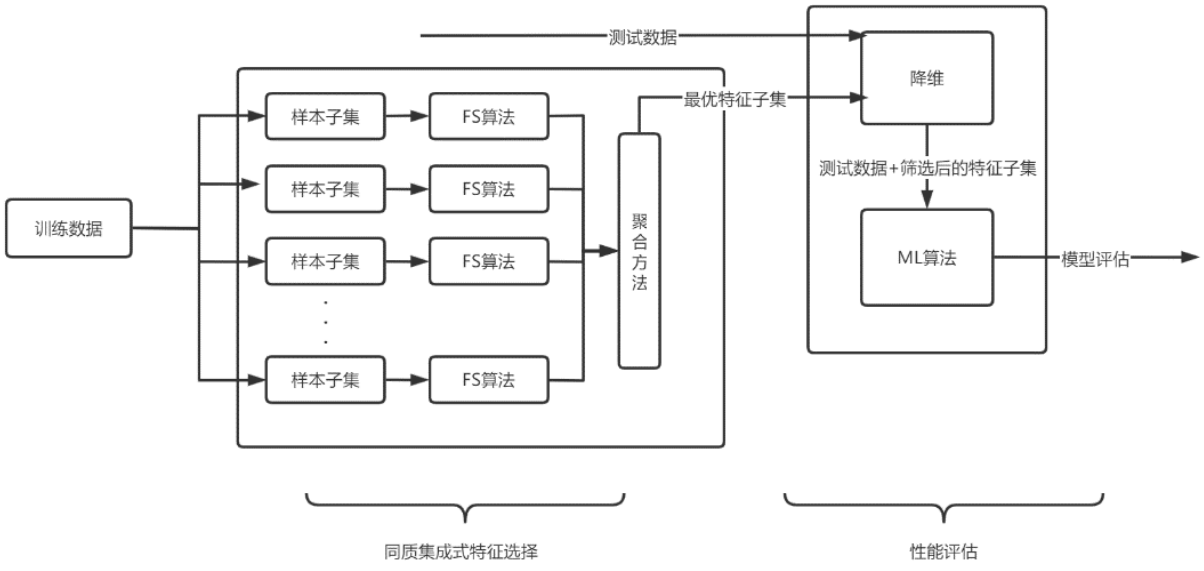


图 11 集成式特征选择模型流程图

5.3.1 基于相关系数的特征筛选模型

传统的相关系数常用来衡量两个数据集合是否在一条线上面，即衡量定距变量间的线性关系，这里，为了衡量变量与目标变量的线性关系，我们使用上节中的 **Spearman 相关系数**计算，特征变量的重要性由该特征与因变量间的相关系数反应，相关系数越大，则该特征的重要性越高。

由于本题中各变量间可能存在非线性关系，传统相关性分析并不足够，为此我们选择**距离相关系数（Distance Correlation）**作为衡量变量间非线性相关性的指标。

Szekely 等人[7]定义了距离相关系数的概念，克服了传统的 Pearson 相关系数的缺点，可以衡量非线性相关变量之间的关系。在某些情况下，即便 Pearson 相关系数为 0，也无法将两个变量判定为线性无关（有可能非线性相关），但是，如果距离相关系数为 0，则可将两个变量判定为互相独立。两个随机向量 $x \in R^P, y \in R^q$ 为观察到的随机样本，x, y 间的距离相关系数(dCor)可以定义为：

$$R^2(x,y) = \frac{v^2(x,y)}{\sqrt{v^2(x,x)v^2(y,y)}} \tag{7}$$

其中：

$$v^2(x,y) = \frac{1}{n^2}\sum_{i,j=1}^{n} A_{i,j}B_{i,j} \tag{8}$$

$$A_{i,j} = \|x_i - x_j\|_2 - \frac{1}{n}\sum_{k=1}^{n}\|x_k - x_j\|_2 - \frac{1}{n}\sum_{l=1}^{n}\|x_i - x_l\|_2 + \frac{1}{n^2}\sum_{k,l=1}^{n}\|x_k - x_l\|_2 \tag{9}$$

$$B_{i,j} = \|y_i - y_j\|_2 - \frac{1}{n}\sum_{k=1}^{n}\|y_k - y_j\|_2 - \frac{1}{n}\sum_{l=1}^{n}\|y_i - y_l\|_2 + \frac{1}{n^2}\sum_{k,l=1}^{n}\|y_k - y_l\|_2 \tag{10}$$

距离相关系数越大，表示两者相关性越强。在统计学中，对相关性强弱有如下约定，如表 3 所示。

表 3 相关程度度量表

相关性	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
极弱或无相关	0~0.2

5.3.2 基于随机森林的特征筛选模型

随机森林（Random forest，简称 RF）是由美国科学家 Leo Breiman 于 2001 年发表的一种机器学习算法[8]。作为一种新兴起、高度灵活的机器学习算法，其拥有广泛的应用前景，在大量分类以及回归问题中具有极好的准确率。并且，随机森林算法自带特征筛选机

制，即随机森林能够评估各个特征在相应问题上的重要性。在本题中，针对分子描述符变量数量多、高度非线性、耦合度高的特点，选择嵌入式特征选择方法随机森林进行数学建模，以期望得到前 20 个对生物活性最具有显著影响的分子描述符变量。

(1) 随机森林训练

随机森林是以K个决策树$\{h(X,\theta_k),k=1,2,...,K\}$为基本分类器，进行集成学习后得到的一个组合分类器。当输入待分类样本时，随机森林输出的分类结果由每个决策树的分类结果简单投票决定。这里的$\{\theta_k,k=1,2,...,K\}$式一个随机变量序列，它是由随机森林的两大随机化思想决定的：

Bagging 思想：从原样本集X中有放回地随机抽取K个与原样本集同样大小的训练样本集$\{T_k,k=1,2,...,K\}$，每个训练样本集T_K构造一个对应的决策树（本文设定K=500）。

特征子空间思想：在对决策树每个节点进行分裂时，从全部属性中等概率随机抽取一个子集（通常取$\log_2(M)+1$个属性, 为特征总数），再从这个子集中选择一个最优属性来分裂节点（本文设定 M=150）。

训练随机森林的过程就是训练各个决策树的过程，由于各个决策树的训练是相互独立的，因此随机森林的训练可以通过并行处理来实现，这将大大提高生成模型的效率。随机森林的训练流程如图 12 所示。

(2)特征重要性计算

随机森林算法可以在分类的基础上进行回归分析，通过将样本分类的结果进行一定的运算可以获得各个特征重要性特征的重要性表示特征对预测结果影响程度，某一特征重要性越大，表明该特征对预测结果的影响越大，重要性越小，表明该特征对预测结果越小。随机森林算法中某一特征的重要性，是该特征在内部所有决策树重要性的平均值，而在决策树中，计算某一个节点 *k* 的重要性:

$$n_k=\omega_k*G_k-\omega_{left}*G_{left}-\omega_{right}*G_{right} \tag{11}$$

其中，$\omega_k,\omega_{left},\omega_{right}$分别为节点 k 以及其左右节点中训练样本与总训练样本数目的比例，G_k,G_{left},G_{right}分别为节点k以及其左右子节点的不纯度。知道每一个节点的重要性之后，即通过公式的得出某一特征的重要性。

$$f=\frac{\sum_{j\in nodes\ split\ on\ feature\ i} n_j}{\sum_{k\in all\ nodes} n_k} \tag{12}$$

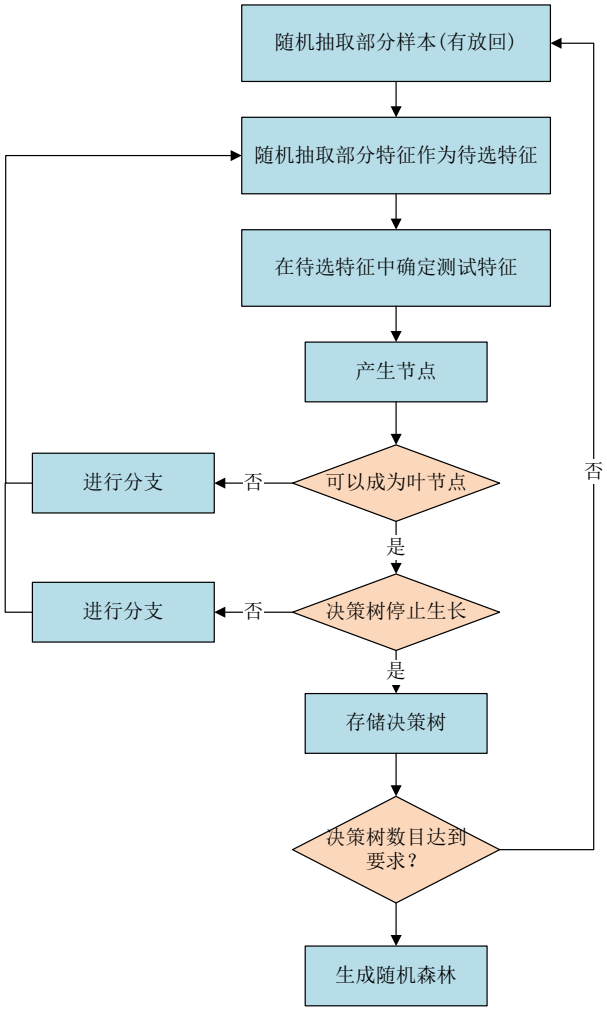


图 12 随机森林训练过程流程图

5.3.3 基于弹性网络的特征筛选模型

弹性网络(Elastic Net，下简称 en)是一种使用 L1，L2 范数作为先验正则项训练的线性回归模型。这种组合可以学习到一个只有少量参数是非零稀疏的模型，就像 Lasso 回归一样，但是它仍然保持一些像 Ridge 回归的正则性质。

弹性网络在很多特征相互联系的情况下是十分有效的。其中 Lasso 回归由于增加了 L1 正则项会产生大量稀疏解，而大量的稀疏解可能会导致最后特征数量不足 20 个而且可能只会考虑两个相关特征中的一个，而弹性网络更倾向于选择两个。所以弹性网络是 Lasso 回归和 Ridge 回归之间的一种权衡，在产生稀疏解的同时继承了 Ridge 回归的稳定性。

其中弹性网络最小化的目标函数是：

$$\min_{\omega} \frac{1}{2n_{sample}} \|X\omega - y\|_2^2 + \alpha\rho\|\omega\|_1 + \frac{\alpha(1-\rho)}{2}\|\omega\|_2^2 \tag{13}$$

其中弹性网络的惩罚函数$\alpha\rho\|\omega\|_1+\frac{\alpha(1-\rho)}{2}\|\omega\|_2^2$中，当$\rho=1$时，弹性网络即为Lasso回归，而当$\rho=0$时，弹性网络即为Ridge回归。其中回归系数$w$即反应特征的重要性。

5.4 四种方法的特征选择结果

采用Python语言和机器学习sklearn包，对四种方法进行编程实现，四种方法分别对剔除无关和冗余特征后的137个变量进行建模。首先，画出各个方法对137个变量重要性的直方分布图，如图13所示，可以看出全部呈正态分布，大部分特征集中较低重要性的特征区域，侧面说明四种方法选出的前二十种特征具有统计学意义的代表性。

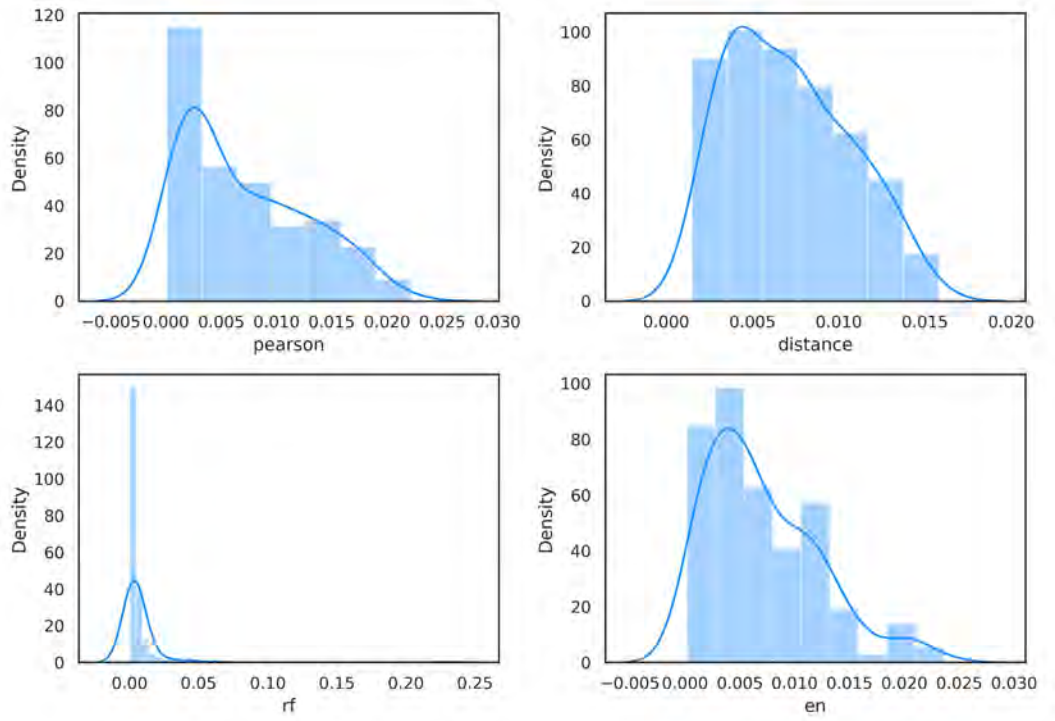


图 13 四种特征选择方法特征重要性分布图

四种方法选出的前20个对生物活性具有显著影响的具体特征见下图。可以发现，使用线性和非线性的相关系数进行特征重要性排序的结果大体一致，而使用RF和弹性网络回归则与前两种方法有一定差别，这是因为RF是通过将样本训练预测的结果进行反推获得各个特征重要性特征的重要性，重要性表示的是对预测结果影响程度，重要性越小，表明该特征对预测结果越小，而弹性网络回归则加入了特征之间相关性的考量。

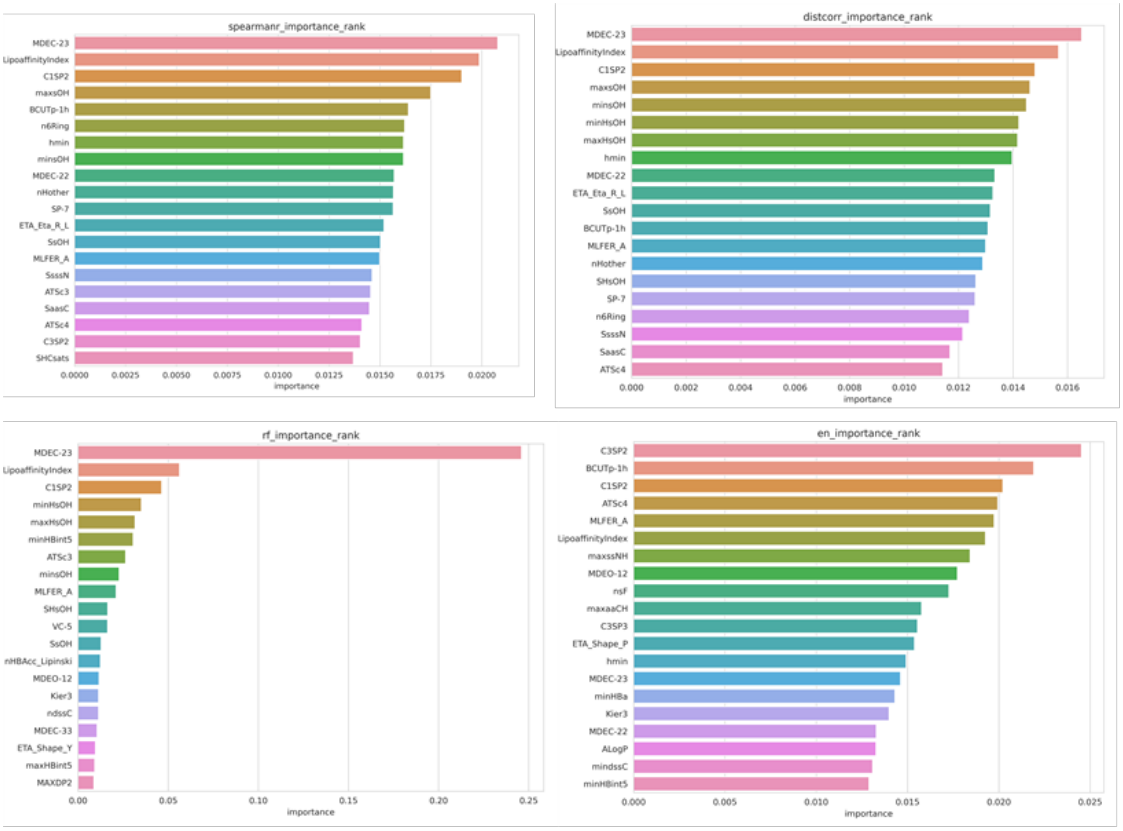


图 14 四种方法筛选出的前 20 个分子描述符变量

5.5 建立集成特征筛选模型

根据上文对四种方法结果的分析，不同的特征选择方法因为模型方法的不同，所选择出的特征变量也有所不同，这里作出四种方法特征选择结果的相关性热力图，如图 13 所示，可以发现，除了两种使用相关性筛选方法的相关性较强以外，大部分具有较弱相关性，利于模型的融合与集成。因此，为了综合这些方法对线性、非线性、高耦合特征的建模能力，基于本题的小样本高维数据，本文建立了集成特征筛选模型，期望获得更加灵活，过拟合风险更低的筛选模型，具体的模型验证将在下一节描述。

四种方法均是以 137 个分子描述符为自变量，生物活性 pIC_{50} 值为因变量建立的数学模型。为了综合四种方法的特征重要性，首先对特征重要性进行归一化，得到特征权值，其反映了变量的重要程度占比，每一个分子描述符的特征权值，表示为该特征的重要性与全体特征重要性之和的比值，其公式如下：

$$y = \frac{y_0 - y_{\min}}{y_{\max} - y_{\min}} \tag{14}$$

归一化后，对每种方法的特征权值进行加权融合，即可得到集成的 137 个自变量的重要性向量：

$$Y = \sum_{i=1}^{4} \lambda_i y_i \tag{15}$$

其中，Y 为集成四个结果的特征重要性向量，λ_i 为第 i 个方法在融合中所占的权重，并满足权重和为 1，y_i 为第 i 个方法归一化后的特征权值。

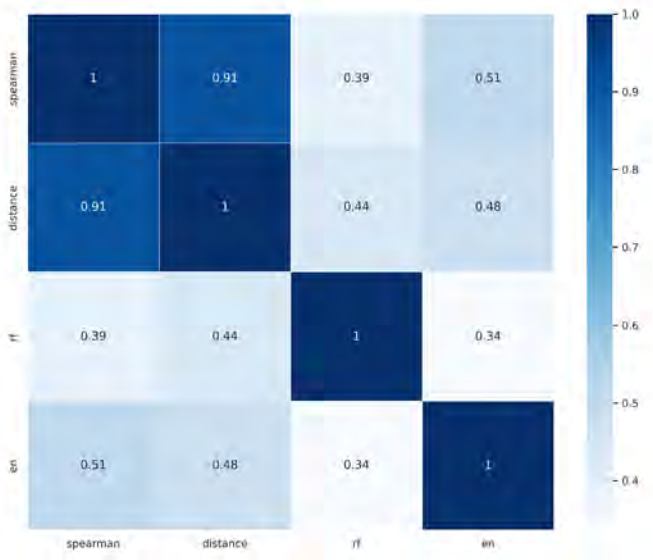


图 15 各特征选择方法的相关性热图

5.6 集成筛选模型的特征选择结果

对建立的集成特征筛选模型进行求解，由此得到的问题 1 的结果，前 20 个对生物活性最具有显著影响的分子描述符（即变量），变量的具体信息见下表。

表 4 集成筛选模型的前 20 个显著变量

排名	分子描述符	重要性	排名	分子描述符	重要性
1	MDEC-23	0.041840	11	C3SP2	0.013543
2	LipoaffinityIndex	0.022608	12	ATSc4	0.013393
3	C1SP2	0.021030	13	nHother	0.013308
4	MLFER_A	0.015666	14	ETA_Eta_R_L	0.013277
5	minsOH	0.015370	15	SP-7	0.013265
6	BCUTp-1h	0.015364	16	SsOH	0.013027
7	maxsOH	0.014963	17	SsssN	0.012685
8	ATSc3	0.014854	18	n6Ring	0.012660
9	hmin	0.014638	19	SaasC	0.012533
10	MDEC-22	0.013979	20	SHCsats	0.012274

通过下图的可视化，可以发现前三 MDEC-23、LipoaffinityIndex、C1SP2 分子描述符的特征重要性较高，后 17 个特征的重要性基本持平。

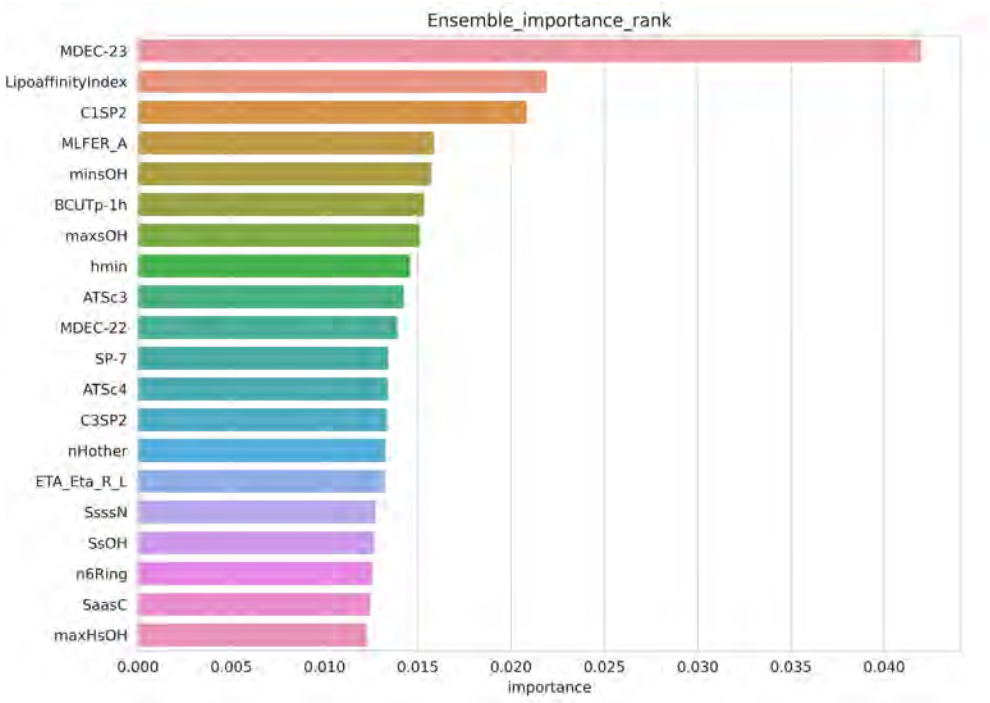


图 16 集成筛选模型的特征选择结果

5.7 显著变量筛选的合理性验证

本小节针对问题 1 提出的分步特征集成筛选模型进行了合理性验证，筛选后的特征变量只有满足独立性和重要性两个方面，才能符合前 20 个变量对生物活性最具有显著影响、即对回归模型的精度有提升的要求。首先，我们对筛选出的 20 个变量进行了相关性分析，证明了提出的模型满足特征变量独立性的要求；其次，我们使用若干个机器学习回归方法对模型进行了详细的消融对比实验，证明了提出的模型满足特征变量代表性的要求。

5.7.1 变量独立性验证

当主要变量两两之间相关性均较弱时 ，某种意义上 任何主要变量均可独立的描述因变量的某方面性质 此时主要变量具有一定的独立性。参照 5.2 节，我们对提出的分步集成筛选模型的结果: 前 20 个显著变量进行了相关性热力图可视化，见下图。可见，变量之间的独立性较好，满足特征变量独立性的要求。

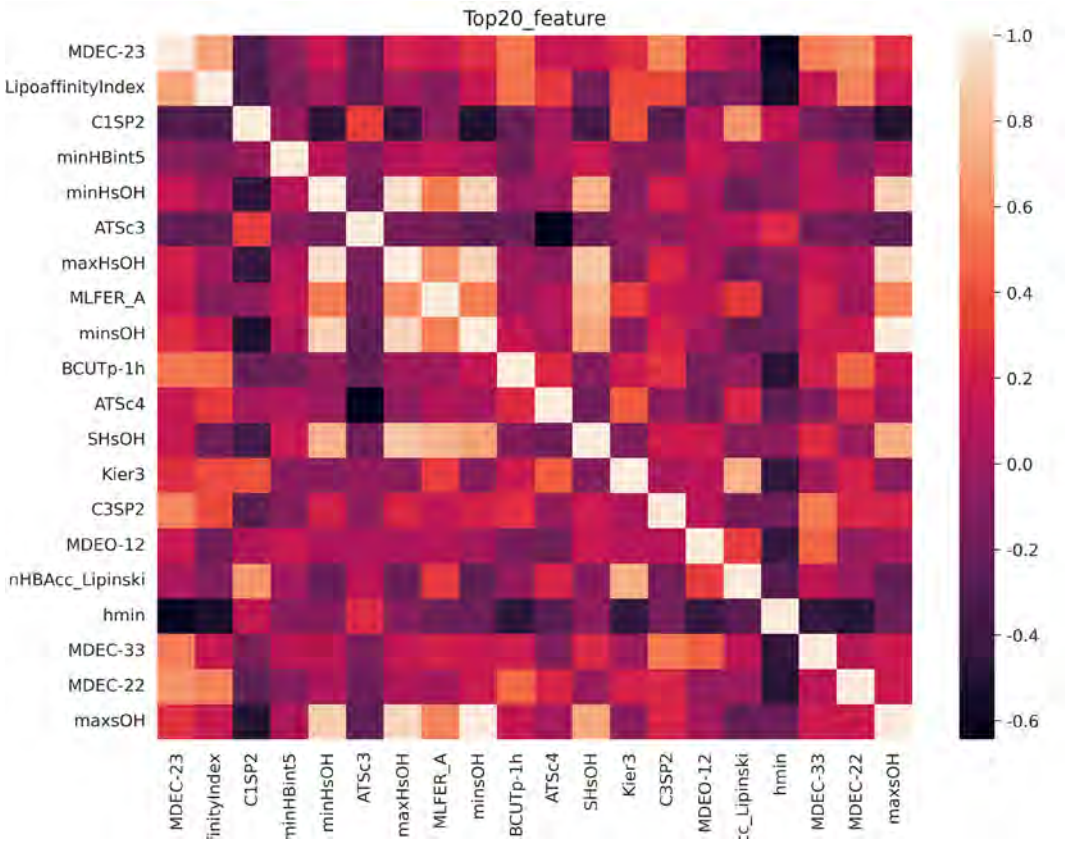


图 17 前 20 个变量的相关性热图

5.7.2 变量代表性验证

首先，我们对筛除冗余特征后的 137 维特征的重要性进行归一化，并对前二十个变量的重要性进行求和，得出累计重要性为 0.32，大于 20 个变量的平均重要性之和 0.14，说明了模型筛选出的 20 个变量满足代表性的要求。

同时，为了分析集成筛选模型与单个模型对回归性能的影响，我们做了如下的消融实验，我们选择了常见的机器学习回归算法 KNN、SVR、RandomForestRegressor、GradientBoostingRegressor、CascadeForestRegressor 以及 LightGBM 方法分别评估了四种特征选择方法的回归性能，采用回归指标 MAE 和 R2 进行评估。

表 5 特征筛选模型的性能对比试验(MAE 指标)

模型	spearman	distcorr	rf	en	ensemble
KNeighborsRegressor	0.536	0.538	0.578	0.570	0.530
RBF SVR	0.586	0.580	0.580	0.571	0.578
RandomForestRegressor	0.551	0.555	0.532	0.551	0.546
GradientBoostingRegressor	0.586	0.583	0.547	0.573	0.575
CascadeForestRegressor	0.513	0.523	0.525	0.530	0.510

LightGBM	0.527	0.551	0.526	0.543	0.540
平均值	0.549	0.555	0.548	0.556	**0.546**

表 6 特征筛选模型的性能对比试验(R2 指标)

模型	**spearman**	**distcorr**	**rf**	**en**	**ensemble**
KNeighborsRegressor	0.737	0.733	0.692	0.702	0.746
RBF SVR	0.694	0.709	0.699	0.709	0.705
RandomForestRegressor	0.722	0.728	0.741	0.734	0.734
GradientBoostingRegressor	0.697	0.695	0.728	0.717	0.703
CascadeForestRegressor	0.762	0.753	0.752	0.748	0.761
LightGBM	0.739	0.727	0.744	0.732	0.737
平均值	0.725	0.724	0.726	0.724	**0.731**

通过实验对比可以发现，不管是在 MSE 指标还是 R2 指标中，通过集成筛选模型选择出的 20 个特征变量对回归模型建模，其平均性能是最佳的，反映了本文提出的分步集成特征筛选模型的有效性。

5.8 结论

(1) 对第四章中去除了无关变量的 341 维分子描述符变量进行分步筛选，逐步去除类内和类间相关性强的冗余特征后，剩余 137 维，使用集成筛选模型后，选择出前 20 个对生物活性最具有显著影响的分子描述符。

(2) 采用 Spearman 相关系数对筛选后的主要变量进行独立性验证，对主要变量建立回归预测模型并进行消融实验，阐明了集成筛选后主要变量的代表性。

六、问题二的求解：深度森林回归模型

6.1 问题分析

本题是问题一的递进，要求选择不超过 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，并用建立的模型对 test 表中的 50 个化合物进行 IC_{50} 值和对应的 pIC_{50} 值预测，根据实际 QSAR 建模经验以及数据分析，IC_{50} 值的波动太大，不利于模型的建立，因此采用 pIC_{50} 来表示生物活性值，两者的换算关系为：

$$pIC_{50} = -log_{10} IC_{50} \times 10^{-9} \tag{16}$$

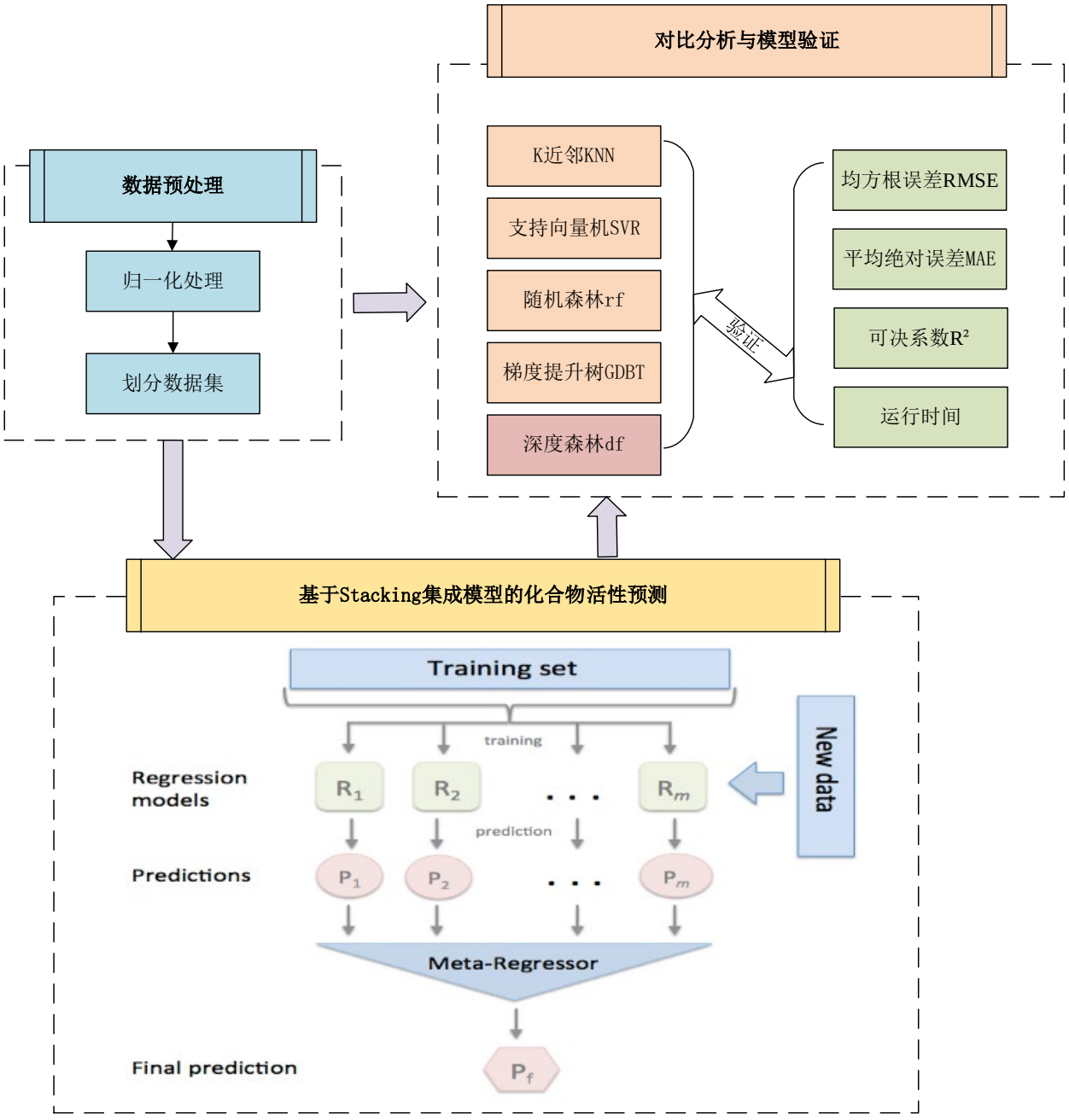


图 18 问题二的模型框架图

根据题意，要求我们使用第一问筛选出的 20 个主要变量利用数据挖掘技术建立化合物生物活性（pIC_{50}）的定量回归模型，并进行模型验证。换句话说，也就是需要我们建立化合物生物活性（pIC_{50}）与主要变量之间的数量关系，然后利用合适的度量评估指标建立数学模型进行验证。本体的难点在于：选择何种的机器学习算法，如何验证该算法对解决本问题的优越性。

首先是预测模型的选取，基于对本题数据的考量，分子描述符的多样化以及本身的复杂性，是使得主要变量之间具有高度非线性，且与因变量生物活性之间的关系较为复杂，因此，若采用线性回归模型可能无法较好地满足建模要求，我们优先考虑采用非线性预测模型来进行建模。

同时，考虑到本题要求使用的变量在 20 个以内，过少的变量难以达到优秀的预测结果，故选择上一问求得的 20 个主要变量。由于特征较少，同时数据样本也不充分，使用深度学习的方法很容易导致模型训练不充分，造成过拟合，且考虑到其计算资源与运行时间，我们放弃使用自动提取特征的神经网络，采用机器学习算法来构建本题的定量预测模型。

最终，本题的模型框架图如图 18 所示，首先对数据进行预处理，并观察了 test 表中 50 个样本与给出的数据样本的分布特征，并划分了数据集；然后，我们分别使用了**集成多种弱分类器的集成(Stacking)模型**和**深度森林(Deep Forest)模型**和来建立回归模型；其次通过实验对两种方法进行对比，阐述两者的优劣；最后，通过验证阐述模型的有效性。

6.2 数据分析与预处理

经过问题一的特征选择工作，我们得到了 1974 个化合物的前 20 个重要分子描述符特征，由于这些分子描述符已假设是由计算机正确计算的，所以无需进行数据清洗。我们只需要我们只需根据数据特征和后续各个模型的数据输入格式进行适当的分析和预处理即可，因此我们主要做了以下工作：

1）训练集与测试集的分布分析

考虑到我们将用在 1974 个化合物数据中建立的模型来预测 test 表中的 50 个样本，因此，我们首先针对两者的特征数据的均值和方法进行了统计，见下表，可见，训练集与无标签的测试集分布较为一致，可以使用 1974 个化合物数据建立的数学模型来预测 50 的样本的化学物生物活性。

分子描述符	mean		std	
	train	test	train	test
MDEC-23	25.58	27.82	9.46	15.77
LipoaffinityIndex	8.67	9.99	3.44	3.65
MLFER_A	0.79	1.35	0.51	1.69
minsOH	0.38	0.42	0.22	0.20
BCUTp-1h	0.21	0.23	0.086	0.064
maxsOH	0.423	0.49	0.24	0.21
...	...	...	...	...
MDEC-22	14.53	17.21	9.10	11.84

2）归一化处理

考虑到数据中有的特征范围波动较大，为了消除不同特征类别差距在数据上的影响，本文将使用的每个输入进行归一化处理，数据归一化后可以方便处理后面数据，保证程序运行加快。其计算公式如下：

$$x_{\text{normalization}} = \frac{x - Min}{\text{Max} - \text{Min}} \tag{17}$$

其中 x 表示变量的值，Min 表示该类变量中的最小值，Max 表示该类变量的最大值，$x_{\text{normalization}}$ 表示归一化的值。

3）数据集划分

对于本题中的所有模型，将所有数据的 70%作为训练集，30%作为验证集。

6.3 基于 Deep Forest 的回归模型建立

Deep Forest 模型是 2017 年周志华教授提出的一个集成森林模型，是传统的决策树模型在广度和深度上的一种集成，和深度学习类似，具有很好的表征学习能力，同时降低了调参难度（超参数规模很小，一套超参数，可以在不同数据集上表现出较好的性能），且在小的数据集上也表现优越[9]。

6.3.1 Deep Forest 模型训练

深度森林模型的训练分为两个阶段：

多粒度扫描阶段：使用多个大小不同的滑动窗口在原始数据特征上进行滑动取值和采样。将原始特征通过多粒度扫描后，可以得到增强的特征向量，包含了特征的局部结构的多种尺度。具体做法如图 18 所示，对于维度为 400 的训练样本，用大小为 100 的滑动窗口扫描，可以得到 301 个滑动样本实例，每个实例维度为 100，公式如下：

$$N = D - W + 1 \tag{18}$$

其中，N 为滑动样本实例，D 为原始特征维度，W 为窗口大小。接着，将 N 个滑动实例样本作为随即森级的输入，并对输出的特征进行拼接，作为接下来模型的输入。可以发现，在多粒度扫描阶段，深度森林采用滑动窗口的方法进行上采样，将一个样本变成 N 个滑动实例样本，这对本题的小样本数据是十分友好的，其不仅还原了原始样本的局部结构，同时也增强了模型的表征能力。

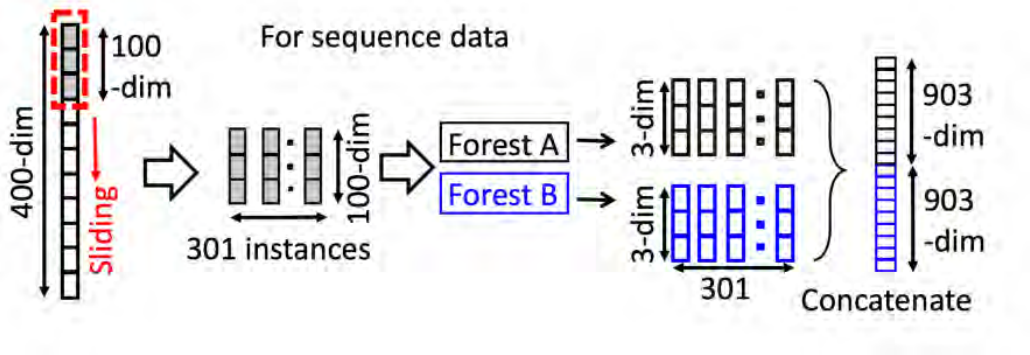


图 19 Deep Forest 模型的多粒度扫描示意图

级联森林阶段：其由图 19 所示的级联随机森林构成，模拟神经网络的分层结构，级联森林每一层使用一组 Forest 来表征，且每一层的森林会使用上一层的信息，作为自己的输入信息，同时其输出为下一层提供输入信息，这是一种表征学习的思想，只是神经网络使用使用卷积或 MLP ，这里使用 Forest。同时，级联森林的输入不仅为上一级的输出，还会与之前的输入拼接，类似与深度神经网络里的残差连接；最后一层森林的输出为 C 个森林输出的平均值，对于本题的回归模型来说，最后输出的为一维。

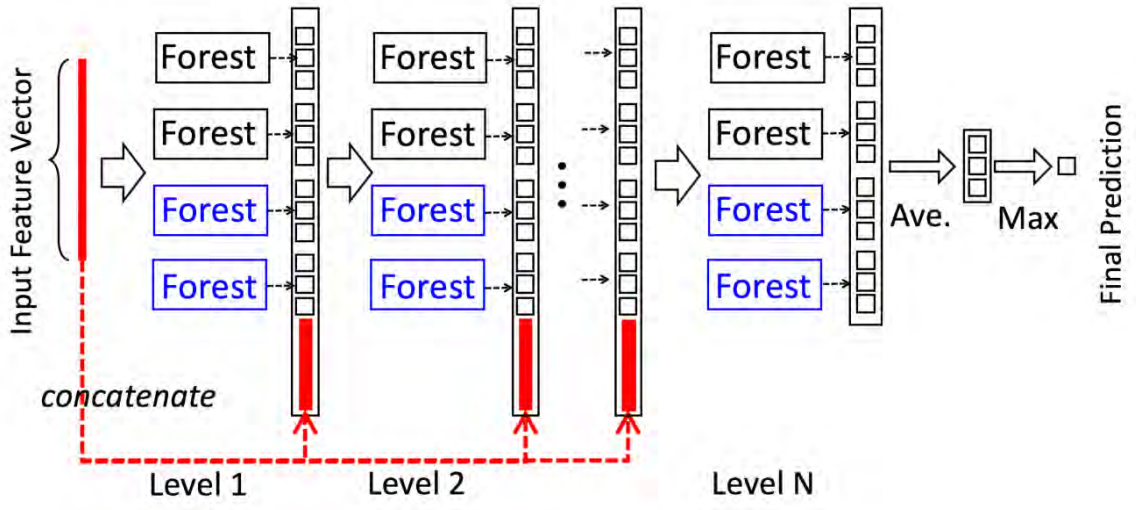


图 20 Deep Forest 模型的级联森林示意图

6.3.2 Deep Forest 参数调优

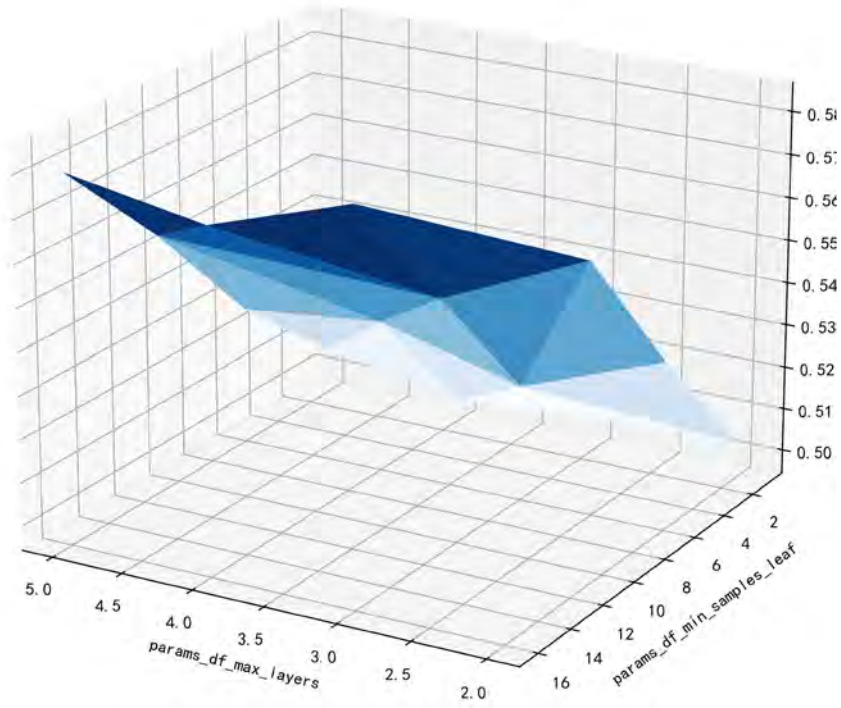


图 21 深度森林模型参数调优图

本节对使用的深度森林算法进行模型优化，使用 Tree Parzen Estimator（TPE）方法进行最佳参数搜索，需要调节的参数为 max_layers、n_trees、min_samples_leaf，对其调节结果如图 21 所示，最终，max_layers、n_trees、min_samples_leaf 分别趋近 3，1，250 时，模型达到最优。

6.4 基于 Stacking 的回归模型建立

Stacking 方法是一种分层模型集成框架，在各种的机器学习任务上有提高模型准确率、增强泛化能力的潜力，其通过组合多个基学习器来完成学习任务。以两层集成为例，首先将数据集分成训练集和测试集，利用训练集训练得到多个初级学习器，然后用初级学习器对测试集进行预测，并将输出值作为下一阶段训练的输入值，最终的标签作为输出值，用于训练次级学习器或元学习器（meta-learner）。由于两次所使用的训练数据不同，因此可以在一定程度上防止过拟合。

Stacking 算法的流程见下表，过程 1-3 是训练出来个体学习器，也就是初级学习器。过程 5-9 是 使用训练出来的个体学习器预测结果，这个预测的结果当做次级学习器的训练集。过程 11 是用初级学习器预测的结果训练出次级学习器，得到我们最后训练的模型。如果想要预测一个数据的输出，只需要把这条数据用初级学习器预测，然后将预测后的结果用次级学习器预测便可。具体地，次级学习器我们在本题中选择线性回归模型，初级学习器的选择见下小节。

表 7 Stacking 集成算法流程

输入：训练集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),...,(\boldsymbol{x}_m,y_m)\}$；
初级学习算法 $\mathfrak{L}_1,\mathfrak{L}_2,...,\mathfrak{L}_T$；
次级学习算法 $\mathfrak{L}$.
过程：
1：**for** t =1,2,…,T **do**
2：$h_t=\mathfrak{L}_t(D)$；
3：**end for**
4: $D'=\varnothing$
5: **for** i =1,2,…,m **do**
6: **for** t =1,2,…,T **do**
7: $z_{it}=h_t(\boldsymbol{x}_i)$
8: **end for**
9: $D'=D'\cup((z_{i1},z_{i2},...,z_{iT}),y_i)$
10: **end for**
11: $h'=\mathfrak{L}(D')$
输出： $H(\boldsymbol{x})=h'(h_1(\boldsymbol{x}),h_2(\boldsymbol{x}),...,h_T(\boldsymbol{x}))$

6.5 对比分析与模型验证

6.5.1 对比方法

为了验证 Stacking 集成算法的有效性，我们比较了以下方法：

- K 近邻回归（KNN）：KNN 回归模型不需要训练参数，只需要借助周围 K 个最近训练样本的目标值，对待测试样本的回归值进行决策。由此就衍生出衡量待测样本回归值的不同方式，即普通的算术平均算法和考虑距离差异的加权平均，本题采用默认的平均方法。
- 支持向量机回归（SVR）：支持向量机回归，顾名思义就是用支持向量机来进行回归预测，而支持向量机是一种按监督学习方式对数据进行二元分类的广义线性分类器，其决策边界是对学习样本求解的最大边距超平面。
- 随机森林回归（rf）：使用随机的方式建立一个森林，森林里面有很多的决策树组成，随机森林的每一棵决策树之间是没有关联的。
- 梯度提升回归（GBR）：梯度提升算法是一种通用的学习算法，除了决策树，还可以使用其它模型作为基的学习器。梯度提升算法的思想是通过调整模型，让损失函数的值不断减小，然后将各个模型加起来作为最终的预测模型。而梯度提升决策树则是以决策树为基的学习器。
- 基于 Stacking 的回归：是一种分层模型集成框架，在本题中，初级学习算法使用以上四种方法，次级学习器使用线性回归模型。
- 深度森林回归(df)：一种集成森林模型，是传统的决策树模型在广度和深度上的一种集成，和深度学习类似，具有很好的表征学习能力，同时降低了调参难度，对小样本数据十分友好。

6.5.2 主要参数说明

设置合适的参数对机器学习方法十分重要，在本题中，为了保证实验结果的客观性和可对比性，本题中用到的模型均采用默认参数。具体参数见下表。

表 8 各个机器学习方法参数表

模型	参数名	参数值
K 近邻回归（KNN）	n_neighbors	5
	weights	'uniform'
	leaf_size	30
	metric	'minkowski'
支持向量机回归（SVR）	kernel	'rbf'
	degree	3
	gamma	'scale'

随机森林回归（rf）	n_estimators	100
	criterion	'squared_error'
	min_samples_split	2
	min_samples_leaf	1
梯度提升回归（GBR）	loss	'squared_error'
	learning_rate	0.1
	n_estimators	100
基于 Stacking 的回归	criterion	'friedman_mse'
	estimators	KNN,SVR,rf,GBR
	final_estimator	LinearRegression
深度森林回归(df)	n_bins	255
	criterion	'mse'
	bin_type	'percentile'
	n_trees	100
	predictor	' forest '
	max_layers	**20**

6.5.3 评价指标

在本小节中，我们采用四个评价指标对各个模型的回归预测结果进行评估：

1）均方误差（MSE）

均方误差（MSE）是指参数估计值与参数真值之差平方的期望值。MSE 是衡量"平均误差"的一种较方便的方法，MSE 可以评价数据的变化程度，MSE 的值越小，说明预测模型描述实验数据具有更好的精确度，其计算公式如下：

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(R_t - P_t)^2 \tag{19}$$

其中，N 表示样本数，R_t 表示 t 样本对应的真实生物活性，P_t 表示 t 样本对应的预测值。

2）平均绝对误差（MAE）

平均绝对误差（MAE）是所有单个观测值与算术平均值的偏差的绝对值的平均，平均绝对误差可以避免误差相互抵消的问题，因而可以准确反映实际预测误差的大小。此外，由于离差被绝对值化，不会出现正负相抵消的情况，其计算公式如下：

$$MAE=\frac{1}{N}\sum_{t=1}^{N}|R_t-P_t| \tag{20}$$

3）可决系数（R^2）

可决系数反映的是预测值对实际值的解释程度。其公式如下

$$R^2=1-\frac{\sum(R_t-P_t)^2}{\sum(R_t-P_t)^2} \tag{21}$$

其中，分母代表原始数据的离散程度，分子为预测数据和原始数据的误差，二者相除可以消除原始数据离散程度的影响。可决系数越接近 1，表明自变量对因变量的解释能力越强，这个模型对数据拟合的也较好，一般来说，$R^2>0.4$，认为拟合效果较好。

4）运行时间（RT）

模型运行时间，即模型从开始训练到完成测试的时间。

6.5.4 实验结果展示及分析

本小节将对各个方法的预测结果进行分析和可视化展示。具体来说，我们分别对上文提出的各个方法进行训练，并在验证集上进行测试，将最终得到的结果与真实生物活性值比较，计算出他们的均方根误差（RMSE）、平均绝对误差（MAE）、可决系数（R^2）以及运行时间（RT）。

为了更直观地对比各个模型的预测结果，我们将验证集样本的编号作为横坐标，对应的化合物生物活性预测值与真实值的残差作为纵坐标，画出了各个模型的残差图，如下图所示，可以发现，K 近邻回归（KNN）、支持向量机回归（SVR）以及梯度提升回归（GBR）模型都对训练集欠拟合，模型的性能不如另外三种模型。

同时，将各个方法在 4 个指标上进行详细地对比，分析出各个模型的验证效果，如下表所示，图 22 绘制了各个模型的训练时间和 MAE 指标的对比图。可总结为以下几点：

1）所有模型的 R2 都大于 0.6，提取的前 20 个重要特征对回归模型有效；

2）四种单模型的预测效果不如基于 Stacking 的回归模型，集成模型的性能远高于单个模型，但是由于单模型的参数少、模型并不复杂，所以训练时间也是相对短的；

3）深度森林回归(df)的效果相较于基于 Stacking 的回归模型不仅在性能上更佳，同时在训练时间上也占优。深度深林回归模型的线性拟合散点图见图 23。

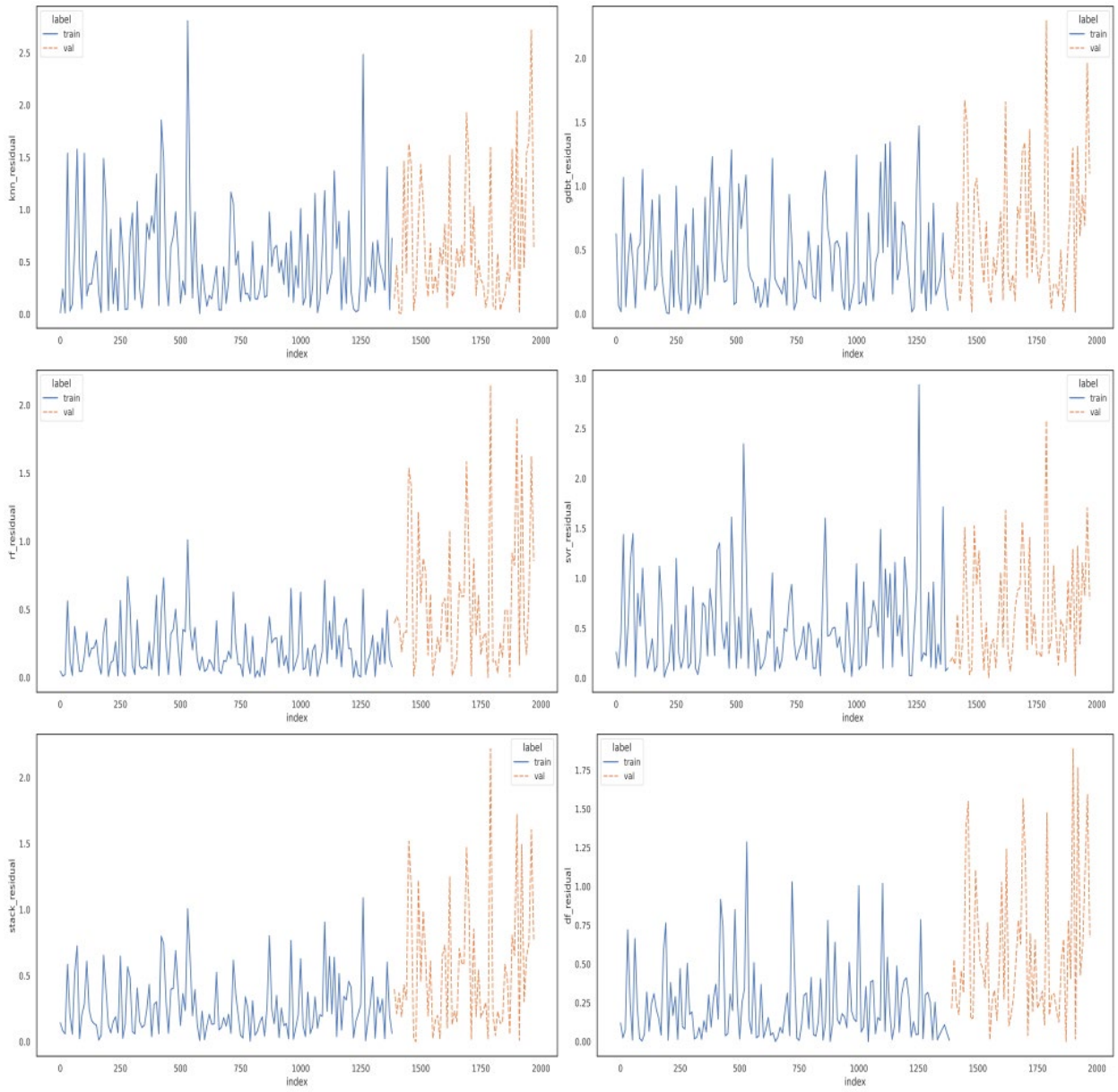


图 22 各个方法的残差图

表 9 各个模型的回归性能对比

模型	MSE	MAE	R2	RT
K 近邻回归（KNN）	0.676	0.594	0.665	0.268
支持向量机回归（SVR）	0.633	0.588	0.686	0.364
随机森林回归（rf）	0.520	0.531	0.742	0.472
梯度提升回归（GBR）	0.589	0.579	0.708	0.614
基于 Stacking 的回归	0.517	0.524	0.744	8.798
深度森林回归(df)	0.476	0.506	0.764	8.31

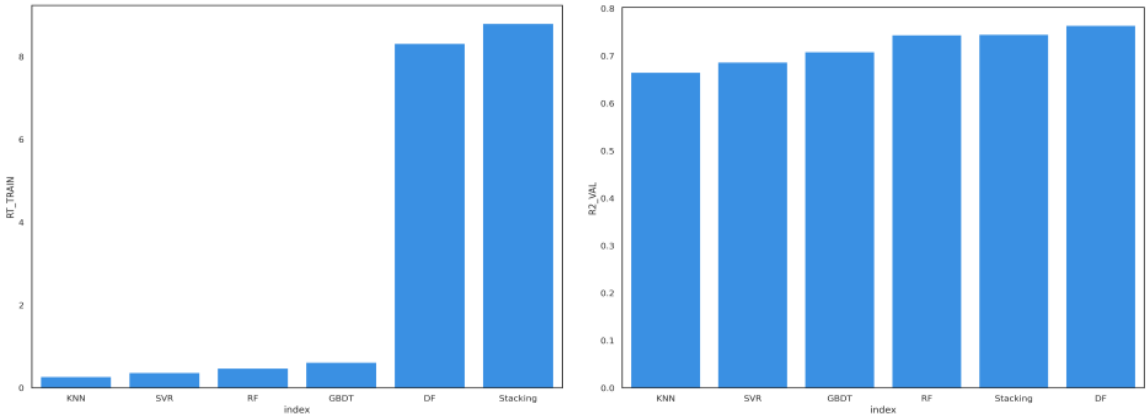


图 23 各个模型的性能对比图

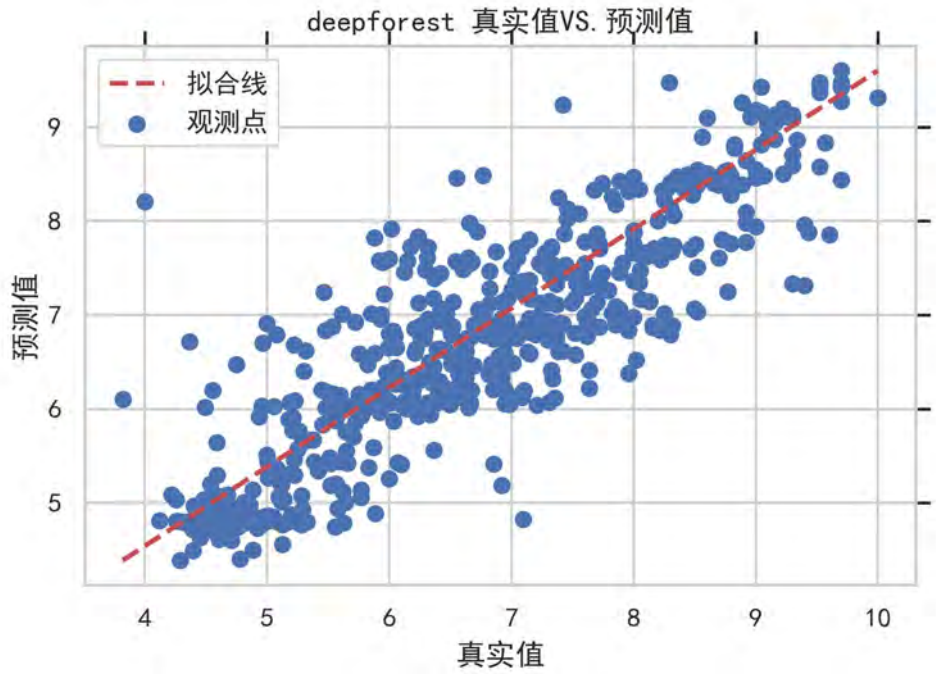


图 24 深度森林回归模型的线性拟合散点图

6.6 结论

1) 观察训练集与测试集的分布，划分数据集 7:3，对问题一筛选出的 20 个特征变量进行建模，采用多种机器学习方法进行模型验证；

2) 经过对比实验与模型验证，深度森林回归(df)的效果和基于 Stacking 的回归模型优于其他模型，同时，深度森林模型在参数优化后 MSE 达到了 0.476，优于其他模型，对本题的生物活性预测有效。

七、问题三的求解：原型网络分类模型

7.1 问题分析

不同于问题二的预测问题，问题三是一个分类问题。根据问题三的要求，需要依据题目提供的 729 个分子描述符变量，对化合物的 ADMET 数据进行建模，ADMET（Absorption 吸收、Distribution 分布、Metabolism 代谢、Excretion 排泄、Toxicity 毒性性质）描述了药物在人体内的药代动力学性质和安全性，其具体含义见下表。题目要求对 5 个因变量分别构建二分类模型，并对 test 表中的 50 个化合物的 ADMET 性质进行预测。

表 10 ADMET 性质说明与含义解释

ADMET 性质	**描述**
小肠上皮细胞渗透性 Caco-2	被人体吸收的能力
细胞色素 P450 酶 CYP3A4	代谢稳定性
化合物心脏安全性评 hERG	心脏毒性
人体口服生物利用度 HOB	口服生物利用度
微核试验 MN	是否具有遗传毒性

对于本题，题目对特征的数量没有要求，且要求建立五个二分类模型，因此构建一个**通用、简单、有效**的模型来解决五个二分类问题是本题的关键。所以本题拟采用特征提取能力强大的**神经网络**建模，其本身具有很强的非线性映射能力，能够学习和构建复杂关系的模型，且网络结构简单，能够通过反向传播算法自动学习有效的特征，并实现高效分类。

一般的神经网络模型对数据量的要求较高，通常大数据才能真正发挥其作用，而近两年流行的一种可以用于 few-shot learning 的**原型网络（Prototypical Networks）**，基于对本题数据量和模型简单通用的特点，本题拟建立基于原型网络的五个二分类模型，并进行模型验证。

7.2 标签观察与分析

本文对 1974 个化合物的 5 个 ADMET 性质的数据分布进行了观察，绘制了如图的 5 中性质的类别分布图：

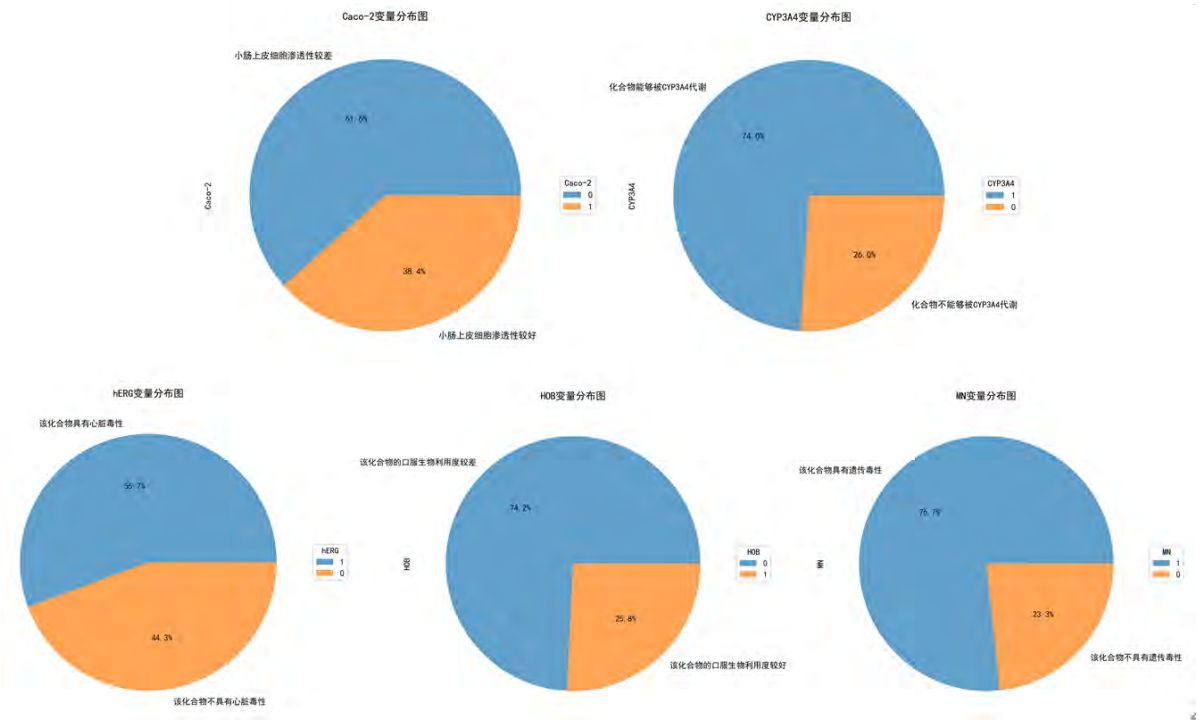


图 25 ADMET 五个性质类别分布图

根据上图我们发现有三种 ADMET 性质的类别分布存在明显的类别不平衡情况，分别为 CYP3A4, HOB, MN。

解决类别不平衡方法大致可以从两个方面入手，一种是从数据层面来解决类别不平衡问题，主要分为欠采样方式和过采样方式，另一种方法是从算法层面来解决类别不平衡问题，主要方法为代价敏感学习方法，下面从数据层面和算法层面主要介绍一下几种方法：

1) 数据层面

- 欠采样方法：直接对训练集中多数类样本进行“欠采样”（under sampling），即去除一些多数类中的样本使得正例、反例数目接近，然后再进行学习，其中比较具有代表性的欠采样算法为 Easy Ensemble，Balance Cascade。
- 过采样方法：对训练集里的少数类进行“过采样”（oversampling），即增加一些少数类样本使得正、反例数目接近，然后再进行学习，比较具有代表性的算法有 SMOTE，Borderline-SMOTE。

2) 算法层面

- 从学习模型出发，对某一具体学习方法的改造，使之能适应不平衡数据下的学习。
- 从贝叶斯风险理论出发，把代价敏感学习看成是分类结果的一种后处理，按照传统方法学习到一个模型，以实现损失最小为目标对结果进行调整。
- 从预处理的角度出发，将代价用于权重调整，使得分类器满足代价敏感的特性。

由于本次化合物数据为小样本数据，比较适合采用过采样方法来解决类别不平衡的问题，因此本文采用了 Borderline-SMOTE 过采样算法来解决类别不平衡问题。

7.3 基于原型网络的模型建立

原型网络（Prototypical Networks）是近年来广泛应用于小样本图像分类的算法。原型网络不同于普通的神经网络，其训练集是由支持集和查询集共同构成，旨在学习到一个嵌入空间使得同类样本在嵌入空间内聚集，不同簇尽可能远离。为了解决小样本学习过程中的过拟合现象，原型网络通过随机抽样可以使有限的标注样本构成远多于样本集数目的学习任务从而避免了算法在训练过程中过拟合。本题中，我们将其扩展到了化合物的 ADMET 性质分类中。

2017 年 Snell 等人[10]首次提出原型网络模型，原型网络是一种简单的元学习模型，算法原理简单但仍能得到良好的训练结果。原型网络的基本思想是从支持集D ={($x1$, $y1$), ⋯⋯ , (x^N, y^N)}中提取每个类原型向量，并根据查询集与每个类的原型向量之间的距离对查询集中的样本点进行分类。更准确地说，原型网络学习了一个嵌入函数 $h(x)$ 用于数据投影，该函数被参数化为神经网络，将各个样本投影到同一空间中，对于每种类型的样本提取他们的中心点(mean)作为原型（prototype），如下图所示，并使用欧几里得距离作为距离度量，使得在该空间中，来自同一类的样本接近，而来自不同类的样本则远离。原型网络的所有参数都位于嵌入函数中。

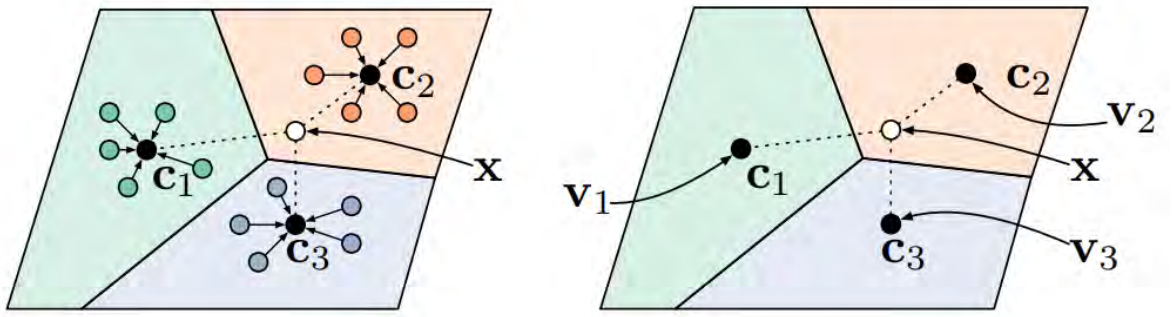


图 26 原型网络示意图

在本文中我们使用基本前向神经网络实现样本数据的嵌入操作，用f_ϕ表示嵌入操作，其中ϕ为可学习参数。原型的计算公式如下。

$$c_k = \frac{1}{|S_k|}\sum_{(x_i,y_i)\in S_k} f_\phi(x_i) \tag{22}$$

其中S_k表示第k类样本。给定查询点（Query point）x ，它所归属的原型的概率分布由基于其与各个原型的距离值得 *softmax* 函数计算得出。

$$p_\phi(y=k|x) = \frac{\exp\left(-d(f_\phi(x),c_k)\right)}{\sum \exp\left(-d(f_\phi(x),c_k)\right)} \tag{23}$$

其中d表示距离计算函数，本文采用平方欧式距离。原型网络的损失函数：

$$loss = -\frac{1}{n}\sum \log\left(p\left(y = c_i^* | x_i^*, \{C_k\}\right)\right) \tag{24}$$

通过在训练中随机抽样得到不重叠的支持集和查询集，使用支持集计算每一类的原型，再通过查询集优化使得 *loss* 收敛。对于全新的样本，经通过查询嵌入点与原型的距离即可得出类别。

7.4 模型求解与验证

7.4.1 模型求解

1）嵌入函数 *h(x)*的选择

对于原型网络，一般使用神经网络来构建嵌入函数，将数据样本投影到同一空间中，从而构建支持集和查询集，本题中，使用了三层神经网络进行建模，关于神经网络的反向传播原理这里不再赘述。我们将第四章中剔除了类内类间冗余特征的 137 个特征喂入神经网络中，利用神经网络强大的非线性特征提取能力，自动提取特征并拟合嵌入函数的参数空间。关于优化器的选择与设置，详见下表：

表 11 原型网络参数设置

名称	参数
优化器	Adam
初始学习率 lr	0.01
weight_decay	0.001
损失函数	CrossEntropy
early_stop	20
max_epoch	500

2）模型优化

下图为原型网络训练过程中的损失曲线以及准确率曲线，在 105epoch 时，训练提前终止，可以观察出，网络的训练过程稳定，能够提取到有效特征，并分类。

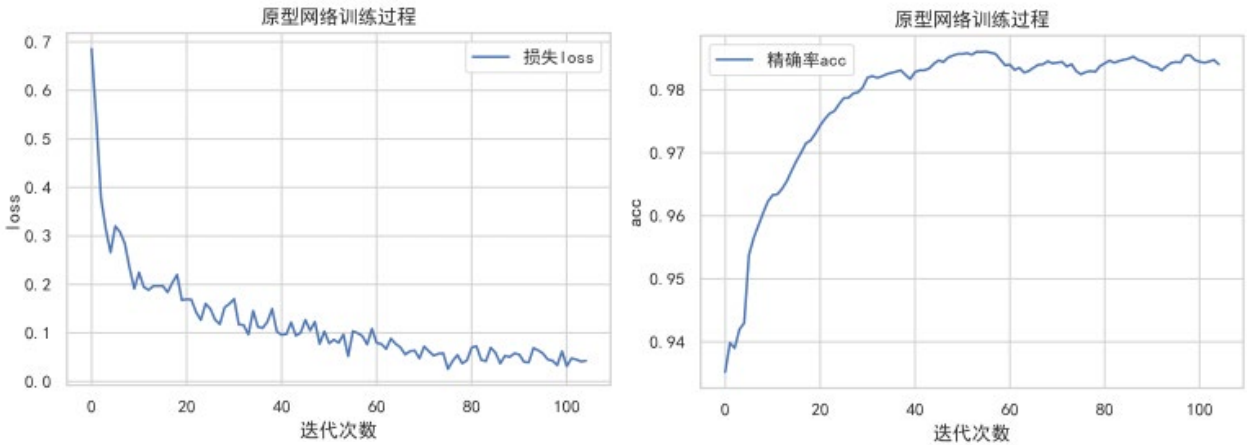


图 27 原型网络的训练过程

模型训练完成之后，将验证集的数据送入参数空间已优化完成的嵌入函数中，得到嵌入空间中数据的分布，见下图所示，散点图已明显分出两个类别，且每个簇的中心也就是原型间的距离较大，分类效果显著。

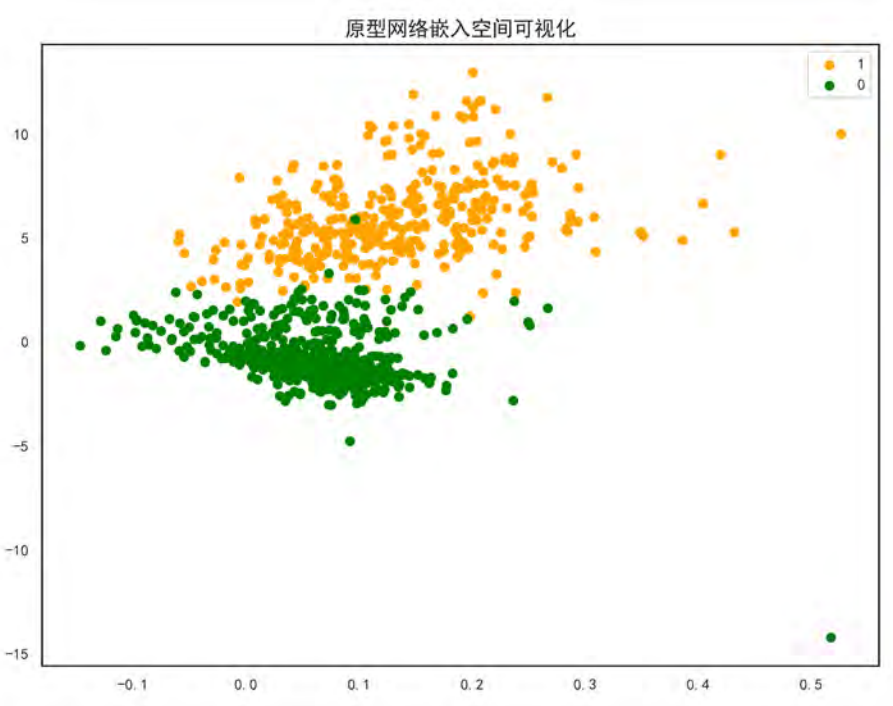


图 28 原型网络嵌入空间

7.4.2 评价指标

1) 准确率（ACC）

准备率是分类问题中最常见的评价指标，一般定义为分类正确的样本数占总样本数的比例，其公式如下：

$$acc = \frac{\text{分类正确的样本个数}}{\text{总样本个数}} \tag{25}$$

但是 **ACC** 指标在数据不均衡的数据集上会缺少一定的说服力，尤其是在有极偏的数据存在的情况下，准确率这个评价指标是不能客观评价算法的优劣的。

2) 混淆矩阵

在判断分类性能好坏时我们一般关心样本归为正确分类的概率、样本归为错误分类的概率是多大，需要参考一系列的分类性能指标，比如召回率、查准率、混淆矩阵等，在多分类问题中比较常用的是混淆矩阵，具体包含内容如下表。通过判断四类情况发生的概率值大小来更好的评价模型分类性能好坏[11]。

表 12 混淆矩阵内容

		预测情况		合计
		1	0	
实际情况	1	实际为 1 预测为 1 概率	实际为 1 预测为 0 概率	实际为 1 概率
	0	实际为 0 预测为 1 概率	实际为 0 预测为 0 概率	实际为 0 概率
合计		预测为正概率	预测为负概率	1

3) AUC 指标

AUC 名为：Area under curve，是指 ROC 曲线下的区域面积。分类器的分类结果一般受分类阈值影响，ROC 曲线考虑了分类器阈值的影响，同时，当测试集中的正负样本分布发生变化了，ROC 曲线仍可以保持不变，有着良好的评估性能。AUC>0.5，则认为分类器优于随机猜测，具有分类效果。

7.4.3 实验结果及可视化

针对 ADMET 的五个性质，我们分别构建了基于原型网络的分类模型，具体性能见下表，可以发现，五个性质的 AUC 性能都超过了 90%，除了人体口服生物利用度 HOB 外，其余四个变量的 AUC 性能均超过 95%，原型网络充分发挥了其在小样本数据集中的优势，详细的 ROC 曲线图和混淆矩阵见图 25，26。

表 13 分类模型的准确率与 AUC 指标

ADMET 性质	**ACC**	**AUC**
Caco-2	0.920	0.974
CYP3A4	0.957	0.987

hERG	0.907	0.965
HOB	0.841	0.906
MN	0.946	0.982

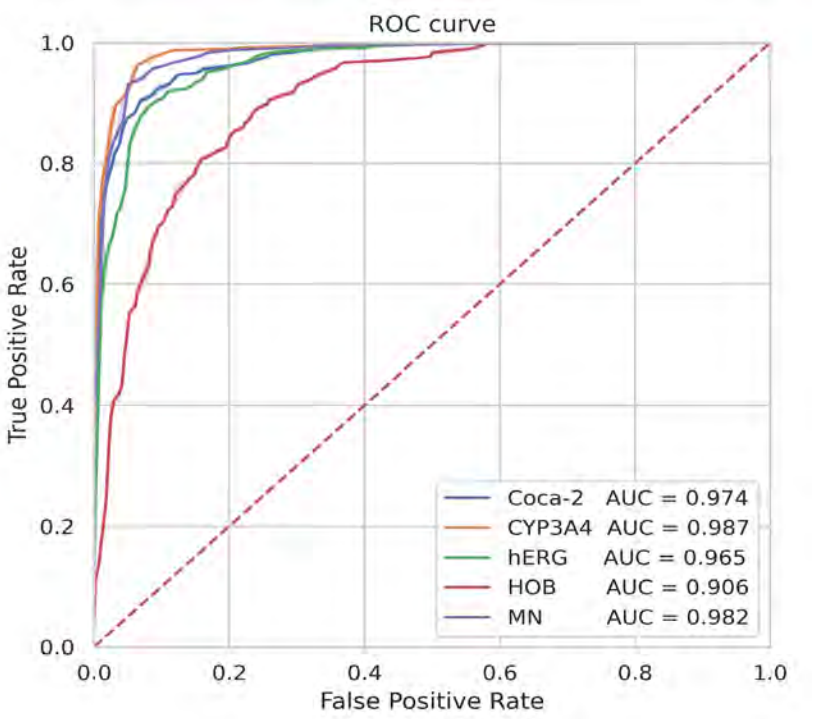


图 29 五个分类模型的 ROC 曲线图

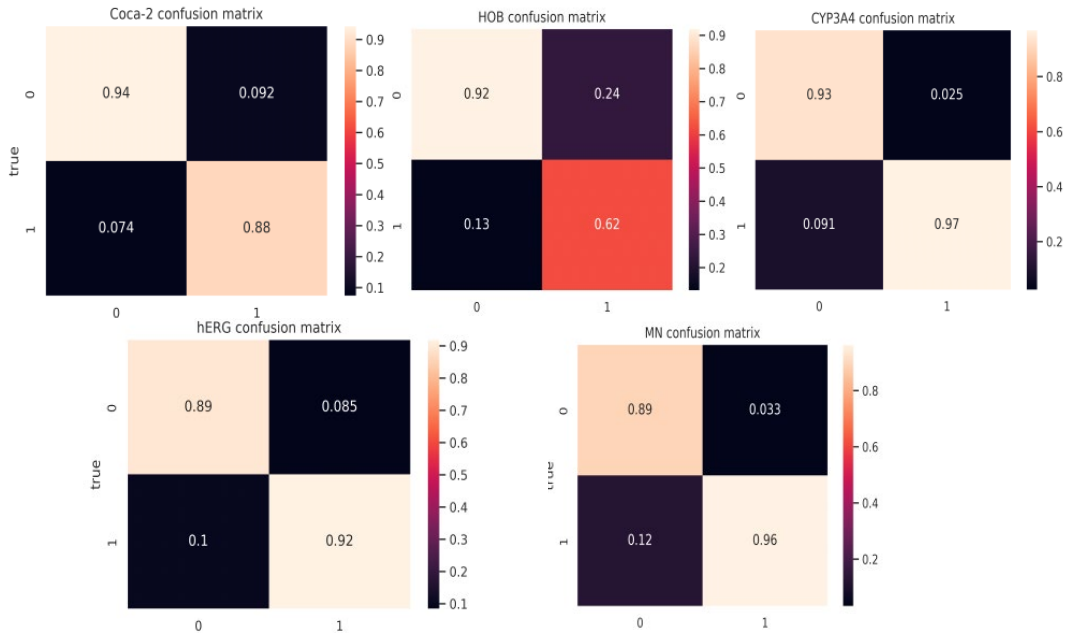


图 30 五个二分类模型的混淆矩阵图

7.5 特征重要性分析

在本题中，我们使用原型网络建立了 137 个自变量与因变量的数学关系，虽然神经网络是一个黑盒模型，我们无法得出神经网络具体的映射函数，但通过使用 MIV 算法，我们仍然能够得到本题建立的原型网络对 137 个自变量的特征重要性排序。

具体公式为：

$$argsort\{\sqrt{\sum_{k=1}^{n}(h(\alpha_i^k)-h(\beta_i^k)^2)}\}_{i=1..N} \tag{26}$$

其中，n 代表嵌入空间的维度，N 代表输入样本的特征维度，这里为 137 维，$h(.)$ 代表优化完成的神经网络参数空间。对增加了同等反向扰动的两个变量α和β，通过神经网络嵌入函数 $h(.)$，得到两个变量在嵌入空间中的位置，进而求出两者在嵌入空间内的欧氏距离，并对所有特征重复此操作，得到 137 维变量的欧氏距离排序，欧式距离越大，说明此特征对因变量的影响越大，即重要性越高。

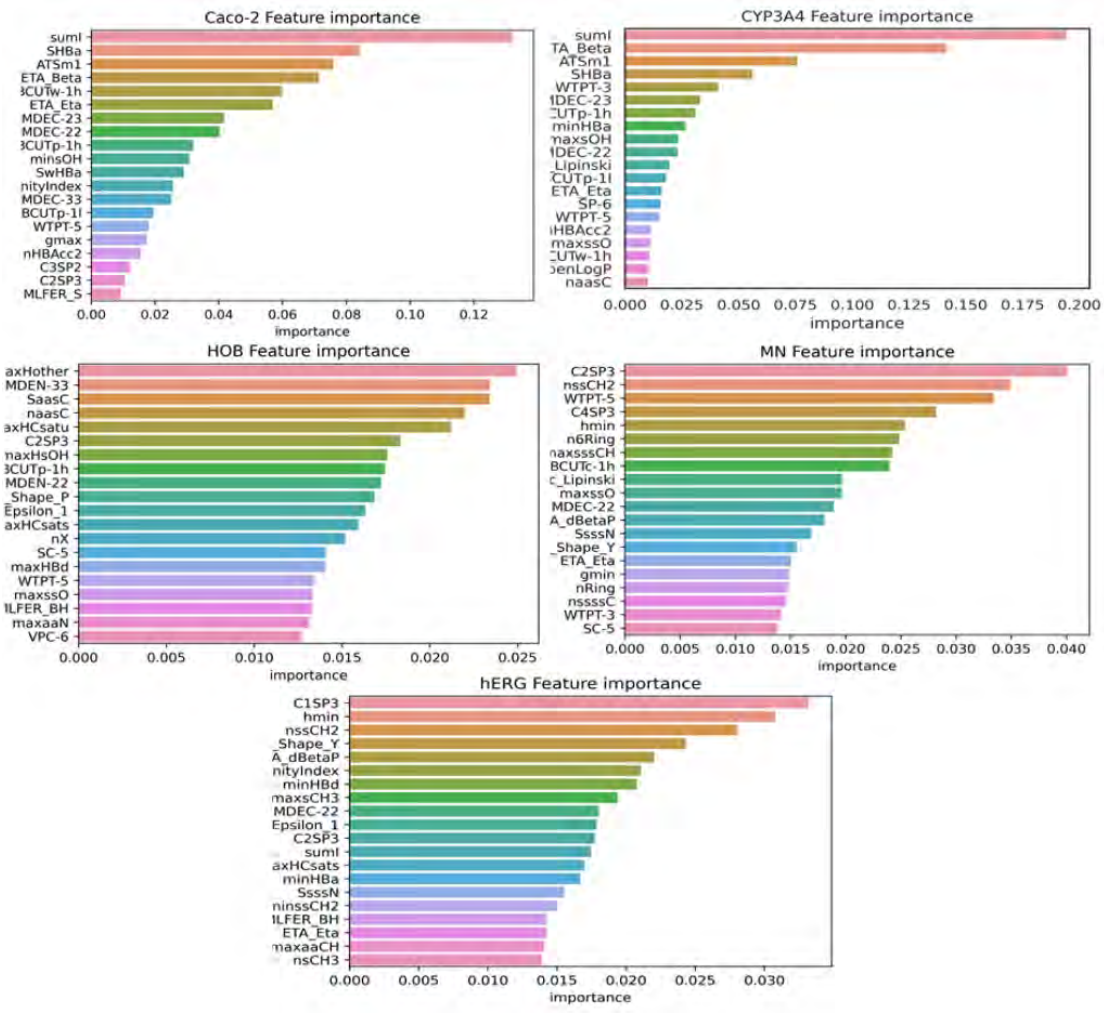


图 31 各个分类模型的前 20 个显著重要特征

我们分别求取了五个模型中对 ADMET 性质分别影响最显著的 20 个变量，特征重要性直方图如图 30 所示。

为了更进一步分析影响五个 ADMET 性质的特征区别与联系，针对 5*20 个可能存在冗余特征的 100 个变量绘制了维恩图，如下所示，经过计算，还剩下 68 个独立特征，这意味着显著影响 5 个 ADMET 性质的分子描述符变量存在一些重复，这些重复存在的分子描述符对化合物的性质有着重大影响，如 MDEC-22, ETA_Eta 两个分子描述符在 4 个分类模型的前 20 个显著特征中都出现，我们将在下一章详细分析。

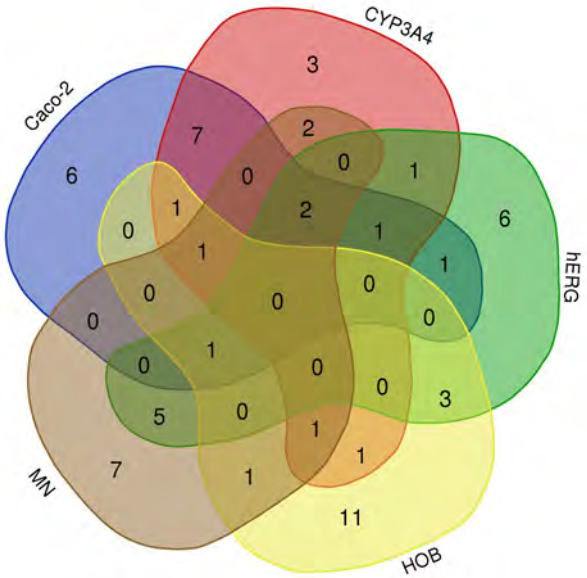


图 32 各个分类模型重要特征维恩图

表 14 各个分类模型重要特征的交集

ADMET 性质	分子描述符交集	数量
Caco-2, CYP3A4, hERG, MN	MDEC-22, ETA_Eta	2
Caco-2,CYP3A4, HOB, MN	WTPT-5	1
Caco-2, hERG ,HOB, MN	C2SP3	1
Caco-2, CYP3A4, hERG	sumI	1
Caco-2, CYP3A4, HOB	BCUTp-1h	1
CYP3A4, HOB, MN	maxssO	1
CYP3A4 Caco-2	ATSm1, BCUTp-1l ,BCUTw-1h ,ETA_Beta ,SHBa, MDEC-23, nHBAcc2	7
CYP3A4, MN	WTPT-3 nHBAcc_Lipinski	2
…	…	…

7.6 结论

1）针对题目要求建立五个二分类模型，本题建立了基于原型网络的分类模型，其具有简单、通用、有效的特点，在特征提取以及小样本训练上展现了优良的性能。

2）通过实验，给出了五个二分类模型的验证效果，实验表明，针对小样本的原型网络在五个 ADMET 性质上都达到了优秀的分类性能；

3）通过 MIV 算法，求解样本在嵌入空间中的欧氏距离，通过对距离的排序逆向得到输入 137 维特征的重要性排序，并使用维恩图对特征进行交叉对比分析，发现对 ADMET 性质具有共性影响的重要分子描述符。

八、问题四的求解：粒子群组合优化模型

8.1 问题分析

根据问题 4 的要求，需要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，根据第二问的分析，我们选取了 20 个特征来预测活性值pIC_{50}的值，根据第三问我们采用了原型网络建立了关于 ADMET 性质的二分类模型，在训练原型网络的时候我们采用了经过冗余特征过滤后的 137 个特征进行训练，并且该 137 个特征包含前 20 个特征，所以我们决定**采用 137 个分子描述符**建立基于初始值优化的粒子群算法的多目标混合整数规划模型来对目标进行优化。

由于这 137 个分子描述符意义复杂且数量巨大，难以通过专家知识来界定这 137 个分子描述符的取值范围，本文采取的做法是统计 1974 个化合物中这 137 个分子描述符的最大值和最小值，然后将这 137 个分子描述符的取值范围约束在这两值之间，并且我们发现 137 个分子描述中存在部分取值只能为整数，比如原子个数，化学键等等，所以我们在初始化种群中对应的整型变量粒子位置时将其**初始化为整数**，并且对这些种群中这些粒子的**速度也设定为整数**，这样就保证了这些**整型变量**优化后的值永远为整数，以上做法就保证了本次优化的合理性。

由于采用的分子描述符的维度比较大，如果采用随机初始化变量方式来优化会导致探索空间十分巨大，容易陷入局部最优，所以本文对初始值的选取进行优化，采用了 1973 个化合物中 ADMET 性质至少有三个较好并且化合物活性pIC_{50}大于 8 的化合物，经过筛选有 35 个化合物满足以上要求，所以本次粒子群算法中的初始种群设定为 70，其中 35 个种群的初始值设定为经过筛选之后的 35 个化合物的分子描述符值，另外 35 个种群的初值则随机生成，这样既保证了**随机性**又融入了**专家经验**，经过实验对比，效**果显著优于完全随机的粒子群优化算法**。

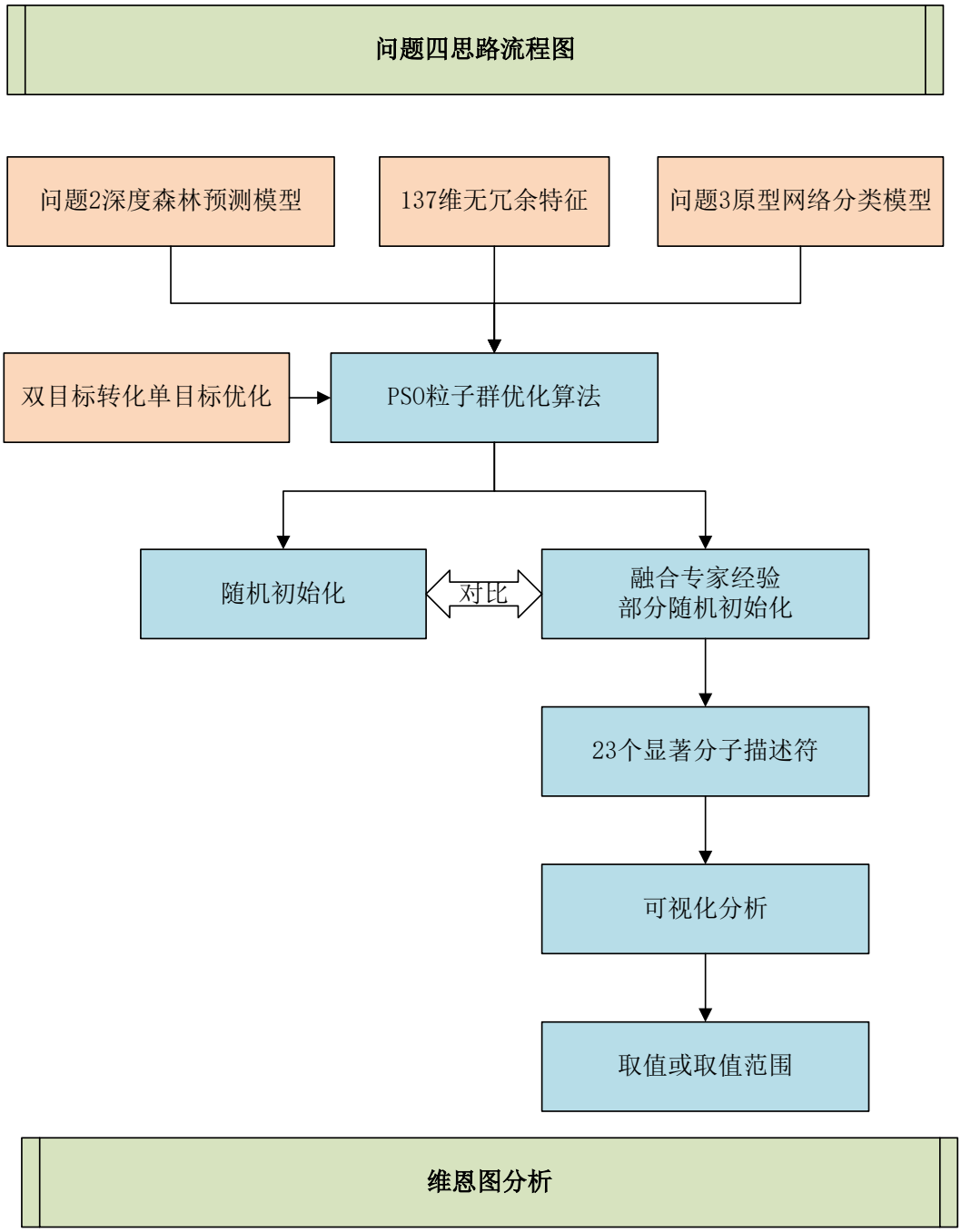


图 33 问题四思路流程图

8.2 化合物分析

经过统计，我们发现 1974 个化合物中 5 个 ADMET 性质中至少有 3 个性质有利于人体的化合物共计有 632 个，5 个 ADMET 性质均有利于人体的化合物有 11 个，如下图所示：

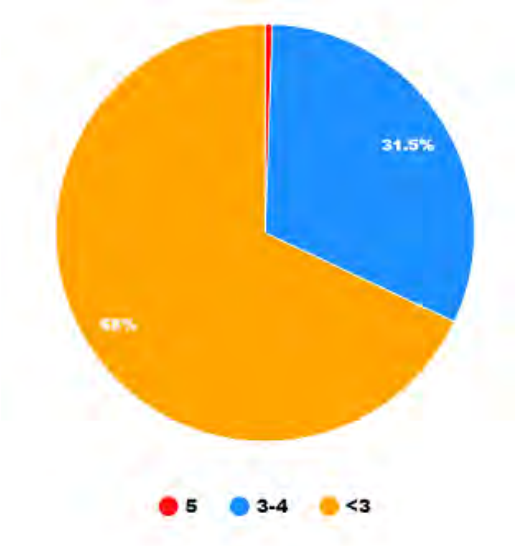


图 34 1974 个化合物的 ADMET 性质分布图

我们对 1974 个化合物中五个 ADMET 性质均为有利于人体且 pIC_{50} 最大的分子进行了筛选，其 pIC_{50} 值为 6.946922，其分子结构图如下：

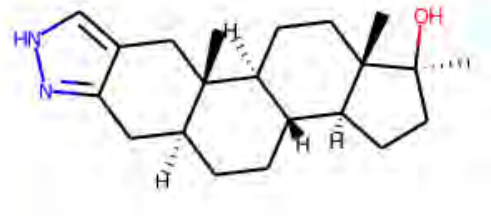


图 35 ADMET 性质最好且活性最高的化合物

同时我们对 1974 个化合物五个 ADMET 性质中有 3 个性质有利于人体且 pIC_{50} 最大的分子进行了筛选，其 pIC_{50} 值为 9.860121，其分子结构图如下：

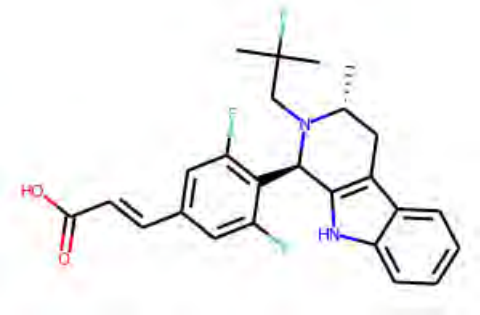


图 36 ADMET 性质满足约束且活性最高的化合物

8.3 基于粒子群 PSO 的优化模型

粒子群优化（Particle Swarm Optimization, PSO），又称粒子群演算法、微粒群算法，是由 J. Kennedy 和 R. C. Eberhart 等于 1995 年开发的一种演化计算技术，来源于对一个简化社会模型的模拟[12]。在该模型中，鸟群的飞行空间即为解空间，鸟群即为粒子群，通过群体信息的共享和更新不断向优化目标方向飞行。

粒子没有质量和体积，只有速度和位置信息，粒子通过速度和位置信息的更新逐渐趋向优化目标，在这个过程中，粒子跟踪个体历史最优位置和群体历史最优位置。粒子群优化算法的迭代示意图如下图所示。

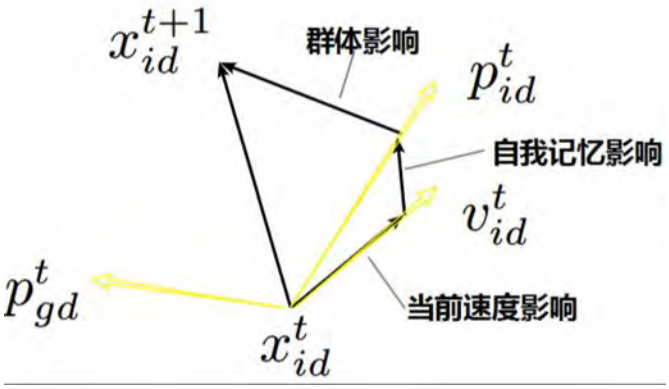


图 37 粒子群算法示意图

假设优化变量的数量为 D，则粒子群的搜索空间为 D 维，在此空间中，有 M 个粒子组成的粒子群。粒子 i 在 t 时刻的位置：

$$x_i^t = (x_{i1}^t, x_{i2}^t, \cdots, x_{id}^t), x_{id}^t \in [l_d, u_d] \tag{27}$$

粒子 i 在 t 时刻的速度：

$$v_i^t = (v_{i1}^t, v_{i2}^t, \cdots, v_{id}^t), v_{id}^t \in [v_{\min}, v_{\max}] \tag{28}$$

其中：$v_{\min}$ 代表粒子速度的最小值；$v_{\max}$ 代表粒子速度的最大值；l_d 代表粒子搜索空间的下限；u_d 代表粒子搜索空间的上限。粒子的每次迭代记录两个位置： 一个是个体最优位置：

$$p_i^t = (p_{i1}^t, p_{i2}^t, \cdots, p_{id}^t) \tag{29}$$

另一个是种群最优位置：

$$p_g^t = (p_{g1}^t, p_{g2}^t, \cdots, p_{gd}^t) \tag{30}$$

其中：$1 \leqslant i \leqslant m$，$1 \leqslant d \leqslant D$。粒子 i 在 $t+1$ 时刻的速度、位置更新公式如下:

$$v_{id}^{t+1}=\omega v_{id}^{t}+c_1 r_1(p_{id}^{t}-x_{id}^{t})+c_2 r_2(p_{gd}^{t}-x_{gd}^{t})$$
$$x_{td}^{t+1}=x_{td}^{t}+v_{id}^{t+1} \tag{31}$$

其中，v_{id}代表在$t+1$时刻迭代粒子i飞行速度矢量的第d维分量；x_{td}代表在$t+1$时刻迭代粒子i位置矢量的第d维分量；c_1、c_2代表学习因子，用以调节学习最大步长；r_1、r_2代表两个的随机数，取值范围$[0,1]$，用以增加搜索随机性； ω 代表惯性权重，非负数，调节对解空间的搜索范围，根据经验，当$w=[0.9,1.2]$时，算法具有比较好的搜索性能. 粒子速度更新公式包含三部分：

1) 分为粒子先前的速度；

2) 第二部分为“认知”部分，表示粒子本身的思考，可以理解为粒子 *i* 当前的位置与自己最好位置之间的距离；

3) 第三部分为“社会”部分，表示粒子间的信息共享与合作，可理解为粒子i当前位置与群体最好位置之间的距离。

8.4 模型建立与求解

8.4.1 建立多目标混合整数规划模型

优化目标：①要求 ADMET 性质中至少有 3 个性质较好，②使得pIC_{50}值尽可能大。

目标函数：采用最优 ADMET 性质-最优活性双目标混合整数规划模型，即同时优化 ADMET 性质以及pIC_{50}的值，目标函数如下：

$$\min\ z_1=-pIC_{50}$$
$$\min\ z_2=-\sum_{i=1}^{5}\mathbb{Z}_{admet}^{i} \tag{32}$$

其中，$\mathbb{Z}_{admet}$为每个化合物的$ADMET$性质的集合，其中 Caco-2，CYP3A4，HOB 均为 1 表示性质较好，而 hERG 和 MN 则为 0 表示性质较好，为了方便后续优化，将 hERG 和 MN 的值都进行取反，定义如下，其中 $\bar{\bullet}$ 表示对 $\bullet$ 进行取反操作。

$$\mathbb{Z}_{admet}=\left\{Caco\text{-}2,CYP3A4,\overline{hERG},HOB,\overline{MN}\right\} \tag{33}$$

决策变量：本文所建立的模型中影响生物活性值主要分子描述符一共有 137 个。因此，该粒子群算法模型中共有 137 个可变变量，即这里的决策变量有 137 个，记为：

$$X=\{x_1,x_2,\cdots,x_{137}\} \tag{34}$$

约束条件：根据问题四要求，在优化pIC_{50}值的过程中需要保证 ADMET 性质中至少有三个较好，这就有了约束 1：

$$\sum_{i=0}^{4}\mathbb{Z}_{admet}^{i}\geqslant 3 \tag{35}$$

同时，通过对分子描述符的取值进行分析，发现了分子描述符存在一定的取值范围，为了设定分子描述符的取值范围，我们对 1974 个化学式的分子描述符的取值进行了统计，将分子描述符的取值范围设定为 1974 个化学式的分子描述符值的最大值和最小值，这样就保证最后优化结果的合理性，于是我们就得到约束 2：

$$lower_i<x_i<upper_i\ \ i=1,...,137 \tag{36}$$

其中，$lower_i$表示x_i的上界，$upper_i$表示x_i的上界。而且我们发现部分分子描述符取值必须为整数，于是我们将 137 个特征中为整型变量的部分筛选了出来，记为：

$$X_z=\{x_z^1,x_z^1,...,x_z^n\} \tag{37}$$

其中n为整型特征的个数，并且在优化过程中需要保证$\mathbb{Z}_z$中每个元素均为整数，这样就得到了约束 3：

$$x_z^1,x_z^2,...,x_z^n\ \ is\ integer \tag{38}$$

在本问中，我们选取的决策变量一共为 137，其中 20 个决策变量为经过集成特征选择选取的重要性排名前 20 的特征，我们将前 20 个特征记为集合：

$$X_T=\{x_T^1,x_T^2,...,x_T^{20}\} \tag{39}$$

其中化合物对抑制$ER\alpha$的生物活性pIC_{50}的深度森林回归模型可由以下非线性函数表示：

$$pIC_{50}=f(x_T^1,x_T^2,...,x_T^{20}) \tag{40}$$

每个化合物的$ADMET$性质的集合的原型网络二分类预测模型可以由以下非线性函数表示：

$$\mathbb{Z}_{admet}^{i}=h_i(x_1\ ,x_2,...,x_{137}),\ i=1,2,\cdots,5 \tag{41}$$

8.4.2 双目标转化为单目标

在本问题中，我们根据最优 ADMET 性质-最优活性双目标混合整数规划模型建立了两个目标函数，选取两个值$(0<\lambda_i<1)$,$i=1,2$ 作为两个目标函数的加权系数，并且$\sum_{i=1}^{2}\lambda_i=1$，由于本问题中，两个目标函数的量纲不同，所以需要采用归一化函数对两个目标函数进行归一化处理，令$\alpha_{z_1}=\frac{z_1-\min z_1}{\max z_1-\min z_1}$，$\alpha_{z_2}=\frac{z_2-\min z_2}{\max z_2-\min z_2}$，其中$\max z_1$，$\min z_1$为 1974 个化合物中最大的$pIC_{50}$的值和最小的$pIC_{50}$的值，$\max z_2$，$\min z_2$为 1974 个化合物中$\sum_{i=1}^{5}\mathbb{Z}_{admet}^{i}$的最大值和最小值，经过归一化处理之后的目标函数变化会变得十分细微，不利于粒子群算法对其进行优化，所以我们将目标函数乘以 100，这样我们就将双目标优化问题转化单目标优化问题，单目标表示如下：

$$\min\ z=(\lambda_1\alpha_{z_1}+\lambda_2\alpha_{z_2})\times 100 \tag{42}$$

因此，该优化模型可以表示为：

$$\min\ z=(\lambda_1\alpha_{z_1}+\lambda_2\alpha_{z_2})\times 100$$

$$\begin{cases}\alpha_{z_1}=\dfrac{z_1-\min z_1}{\max z_1-\min z_1}\\ \alpha_{z_2}=\dfrac{z_2-\min z_2}{\max z_2-\min z_2}\\ z_1=-\,pIC_{50}\\ z_2=-\sum_{i=1}^{5}\mathbb{Z}_{admet}^{i}\end{cases} \tag{43}$$

$$s.t.\begin{cases}\sum_{i=0}^{4}\mathbb{Z}_{admet}^{i}\geqslant 3\\ lower_i<x_i<upper_i\ ,\ x_i\in X\\ z\ is\ integer\ ,\ \ z\in\mathbb{Z}_z\end{cases}$$

8.4.3 基于粒子群算法的单目标混合模型求解

经过初始值优化的粒子群优化算法的流程可总结如下：

Step1：初始化粒子群体（群体规模为 70），将其中 35 个种群的初始位置选取采用了 1973

个化合物中 ADMET 值至少有三个较好并且 pIC_{50} 的值大于 8 的化合物信息，其余 35 个种群的初始位置随机选择，所有种群的初始速度随机初始化，并且对每个种群中的整型变量的初始位置和初始速度约束为整数。
Step2：根据适应度公式，计算每个粒子的适应值;
Step3：对每个粒子，将其当前适应值与其个体历史最佳位置对应的适应值作比较，如 果当前的适应值更高，则判断粒子的当前位置是否满足约束条件，如果满足则将用当前位置更新历史最佳位置，否则不进行更新；
Step4：对每个粒子，将其前适应值与全局最佳位置对应的适应值作比较，如果当前的适应值更高，则将用当前位置更新全局最佳位置;
Step5：根据公式更新每个粒子的速度与位置;
Step6：判断是否满足结束条件，如未满足，则返回 Step2。（通常算法达到最大迭代次数 Gmax 或者最佳适应度值的增量小于某个给定的阈值时算法停止。）

8.4.4 模型参数设定

依据题目要求，选用带约束初始值优化的粒子群进行优化，结合相关因素，粒子群算法的主要变量设定值如下表所示：

表 15 粒子群优化算法参数设置表

要素名称	符号	值
种群大小	m	70
维数	D	137
权重因子	ω	0.8
学习因子	c_1、c_2	1.2
第 i 个粒子的速度范围	$[v_{\min}^i, v_{\max}^i]$	$[lower_i, upper_i]$
最大迭代步长	t	150
初始化各粒子的位置	x_0	随机数+专家经验
初始化各粒子的速度	v_0	随机数
目标函数的加权系数	λ_1 、λ_2	0.7 、0.3

8.5 结果分析与可视化

本次分别采取了全部随机化初始种群的粒子群优化和融合专家经验的部分随机初始化粒子群优化算法进行了对比，实验结果如下图所示：

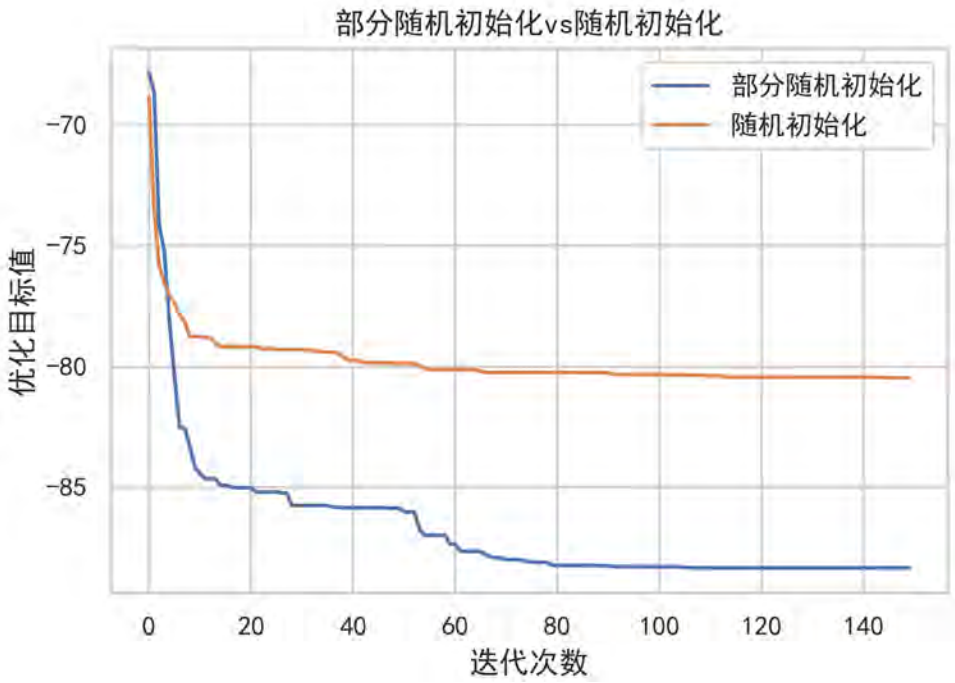


图 38 粒子群优化算法对比实验

通过上图可以明显发现融合专家经验的部分随机初始化的粒子群优化算法优化效果显著优于全部随机化初始种群的粒子群算法。为了验证本题提出的组合优化模型的有效性，我们列出了本文提出的优化粒子群算法中最优个体（150 轮迭代中每一次迭代中的最优个体，共有 36 个不同个体）的迭代过程，即代表了整个全局最优的变化过程，如下表所示，详细优化过程见附件。

表 16 粒子群算法的优化过程

轮次	活性 pIC_{50}	Caco-2	CYP3A4	hERG	HOB	MN
		1 优	1 优	0 优	1 优	0 优
1	7.119955	0	1	0	1	1
2	7.229707	0	1	0	1	1
3	7.165157	0	1	0	1	0
4	7.317654	0	1	0	1	0
5	7.451283	0	1	0	1	0
…	…	…	…	…	…	…
31	8.354529	1	1	0	1	0
32	8.356145	1	1	0	1	0

33	8.357257	1	1	0	1	0
34	8.357596	1	1	0	1	0
35	8.361639	1	1	0	1	0
36	8.368985	1	1	0	1	0

随着粒子群算法的迭代，**五个 ADMET 性质均优化为对人体有益**，并且 pIC_{50} **活性值达到 8.368985**，相比于原始数据样本中，五个 ADMET 性质均对人体有益时 pIC_{50} 活性值仅为 6.946922，得到了较大提升。

在迭代过程中，我们记录了每轮迭代中种群全局最佳位置，并分析了每个分子描述符变量的变化，绘制了如下的部分分子描述符的小提琴图：

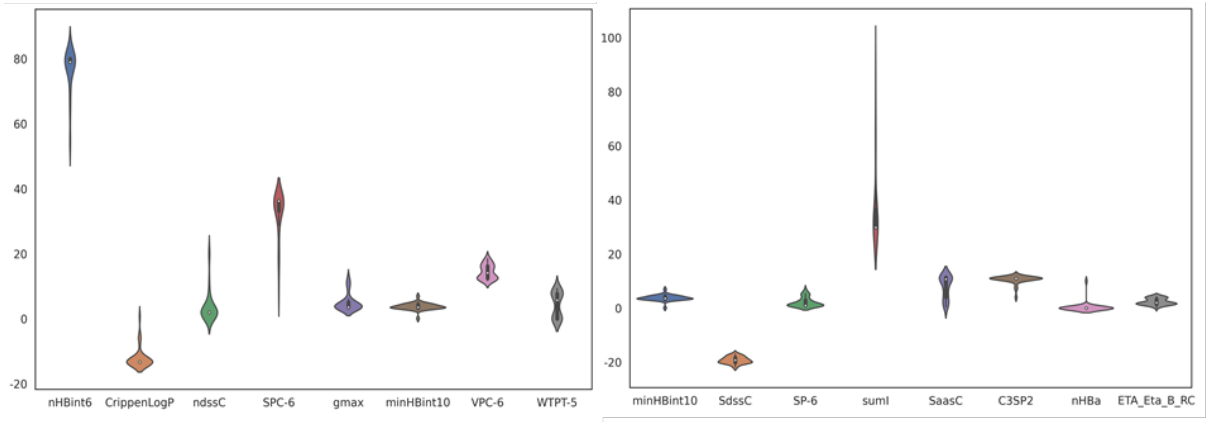


图 39 优化过程中部分分子描述符分布的小提琴图

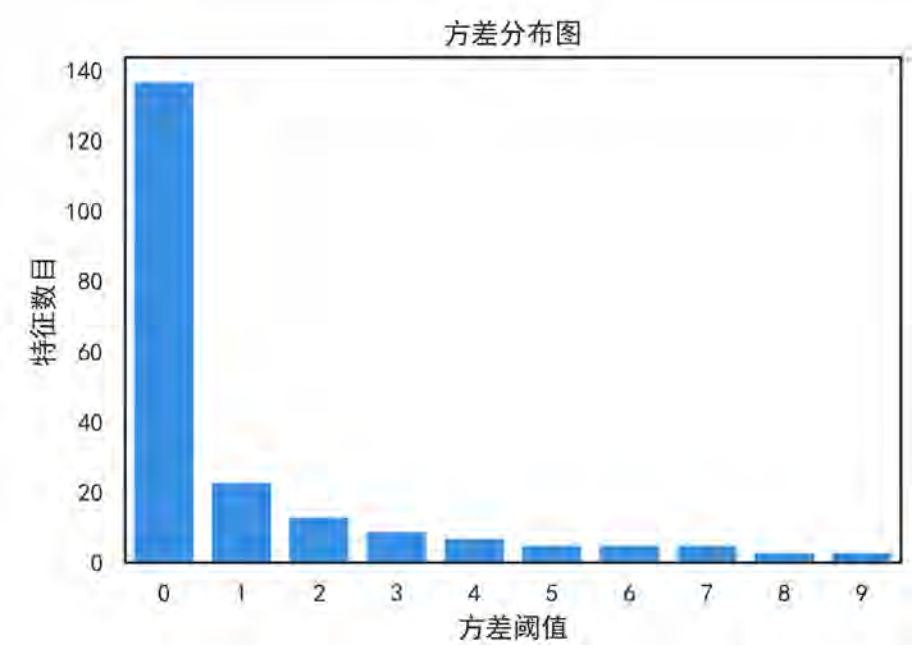


图 40 优化过程中分子描述符的方差阈值图

通过分析，许多特征在优化过程中的波动很小，对于变量波动我们使用优化中的分子描述符方差进行衡量，进而，我们绘制了统计分子描述符变量的方差分布图，如图 39 所示。

从上图可以发现大部分分子描述符在粒子群优化后总体的方差小于 1，没有发生明显变化，这代表这些分子描述符的变化对目标函数的优化没有影响，因此我们将方差小于 1 的分子描述符进行了过滤，最终得到了 23 个分子描述符，我们分析了 150 轮迭代中每一次迭代中的最优个体，其中共有 36 个不同个体，这些个体代表了整个全局最优的变化过程。我们绘制了 23 个主要分子描述符随着活性与 ADMET 性质的变化趋势图：

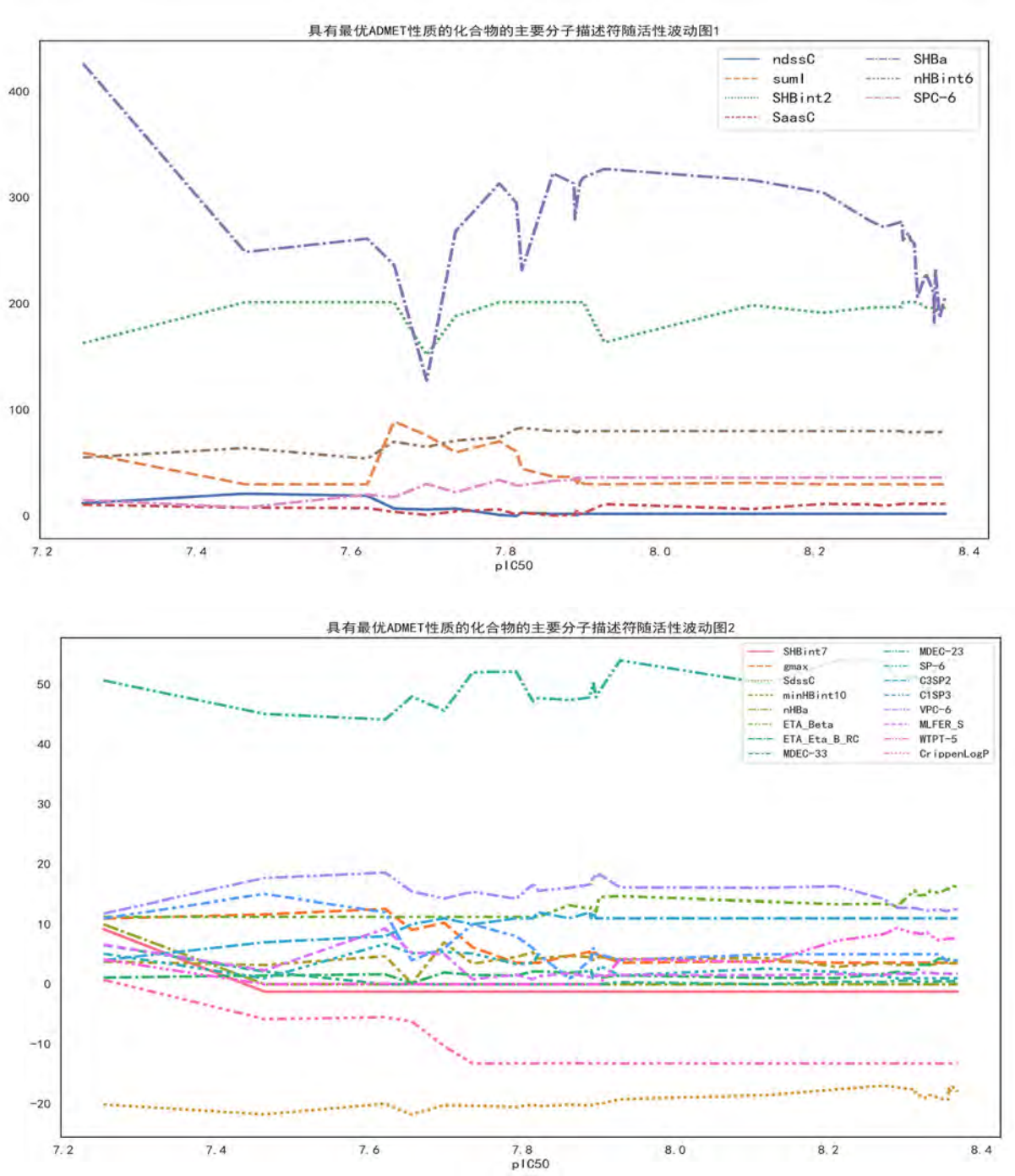


图 41 具有最优 ADMET 性质的化合物的变量随活性波动图

通过统计 1974 个化合物中 5 个 ADMET 性质至少有 3 个较好的pIC_{50}值，其中 95%的化合物样本的pIC_{50}值小于 8，因此我们认为pIC_{50}大于 8 即满足具有更好的生物活性的要求。所以我们统计了所有优化到pIC_{50}值为 8 以上且 5 个 ADMET 性质均对人体有益时的分子描述符变量的最大值和最小值，也就得到了这些分子描述符的取值或取值范围，如下表所示：

表 17 23 个主要分子描述符的取值或取值范围

分子描述符	取值或取值范围	分子描述符	取值或取值范围
SHBint7	[-1.23,-1.22]	ETA_Eta_B_RC	[1.02,4.26]
gmax	[3.58,3.83]	MDEC-33	[0.005,0.566]
ndssC	2.0	MDEC-23	[48.52,54.02]
SdssC	[-19.24, -16.96]	Sp-6	[0.977,2.608]
suml	[29.83,31.18]	C3SP2	11.0
SHBint2	[193.42,201.42]	C1SP3	[4.0,5.0]
minHBint10	[2.90,4.36]	VPC-6	[12.17,16.02]
nHBa	0	SPC-6	[36.09,36.24]
SaasC	[6.58,11.3732]	MLFER_S	[1.49,1.90]
SHBa	[182.26,207.70]	WTPT-5	[3.64,9.36]
nHBint6	[79.0,80]	CrippenLogP	[-13.27,-13.25]
ETA_Beta	[13.0,16.269]		

8.6 与前三问的联系

本题使用粒子群算法得到了 137 个无冗余分子描述符变量的优化曲线，通过组合优化模型与数据分析得出了显著影响化合物生物活性和 ADMET 性质的 23 个主要变量。为了进一步探讨这 23 个分子描述符变量与前三个问题的联系，我们将这 23 个分子描述符、问题一筛选出的前 20 个显著特征、问题三中显著影响五个 ADMET 性质的特征进行了交集验证并绘制了维恩图，如下图所示，其中 Q1 代表问题 1 中筛选出的 20 个变量以及问题 2 中回归模型使用的变量，Q3 代表问题 3 显著影响五个分类模型的主要变量，Q4 代表问题 4 中组合优化模型得出的显著分子描述符变量。

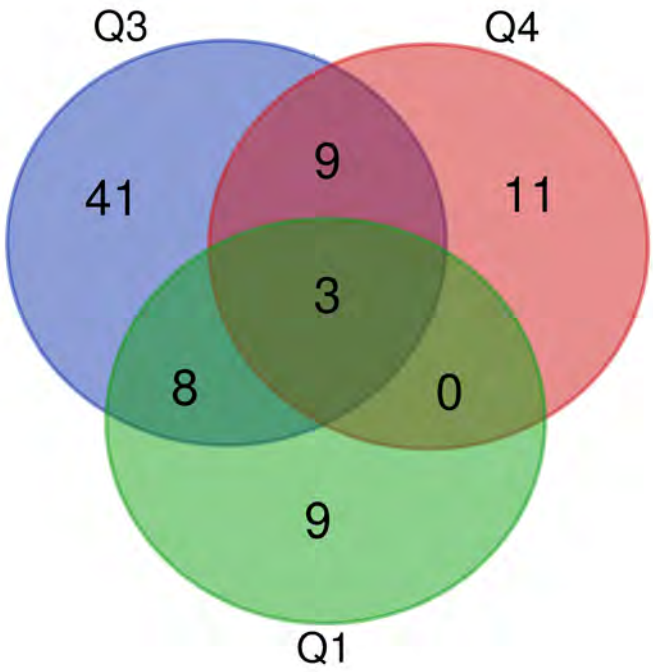


图 42 四个问题间分子描述符的维恩图

可以发现，有 3 个分子描述符变量同时出现在了四个问题中的特征中，这三个特征不仅对定量回归预测 pIC_{50} 活性的模型有显著影响，也对 ADMET 性质分类模型具有重要作用，同时，在组合优化中，这三个特征对优化目标的影响也同样显著。以上结果侧面证明了本文对四个问题建立模型的有效性。经过资料查找，**MDEC-23 描述符**代表分子内所有二级与三级碳分子之间的距离边缘，**SaasC 描述符**代表：：C-原子型电子状态的总和，**C3SP2 描述符**代表双重约束碳与另外三种碳结合。

表 18 四个问题间分子描述符的联系描述

分子描述符	描述	分子描述符	描述
MDEC-23	**Q1∩Q3∩Q4**	WTPT-5	Q3∩Q4
SaasC	**Q1∩Q3∩Q4**	ETA_Beta	Q3∩Q4
C3SP2	**Q1∩Q3∩Q4**	SHBa	Q3∩Q4
C1SP3	Q3∩Q4	gmax	Q3∩Q4
MLFER_S	Q3∩Q4	CrippenLogP	Q3∩Q4
MDEC-33	Q3∩Q4	VPC-6	Q3∩Q4

8.7 结论

1) 针对问题四，我们提出了一种基于初始值优化的粒子群算法的多目标混合整数规划模型来对化合物的pIC_{50}和 ADMET 性质进行组合优化，并且筛选出了 23 个分子描述符变量在优化过程中对pIC_{50}值的影响较大，并给出了这 23 个分子描述符的最优取值范围。

2) 但是由于我们采取的是组合优化，综合考量了pIC_{50}和 ADMET 性质，所以最后优化出来的化合物活性pIC_{50}不如只对 ADMET 三个性质好进行约束优化出的pIC_{50}活性值，但是结果明显优于原始数据中 1974 个化合物中 5 个 ADMET 性质都好的最优活性化合物。

九、模型评价与改进

9.1 模型优点

1) 对于问题一分子描述符的筛选和重要性排序，采用分步集成特征选择算法，融合多种不同的特征选择算法，利用集成算法的优势避免单个算法不稳定的缺点。

2) 对于问题二化合物生物活性的定量预测模型，深度森林模型作为树模型的集成，拥有比 Stacking 更好的回归效果，可以在更短的时间内训练完成，对验证集指标优秀。

3) 对于问题三针对化合物 ADMET 性质的二分类模型，我们选择了先进的原型网络，其利用了神经网络的拟合能力，并具有处理小样本的能力。最后通过 MIV 算法反向推导出了分类过程中的关键变量。

4) 对于问题四，本文改进了粒子群算法，优化了其初始种群的生成过程。达到了同时优化化合物生物活性和 ADMET 性质的目的，在 ADMET 性质不仅达到最优的条件下，而且提升了化合物的生物活性值。

9.2 模型缺点

1）问题一的分步集成特征选择算法虽然具有集成算法的优势，但对于基学习器的选择可能并不是最优，斯皮尔曼系数和距离相关系数的特征筛选结果存在相关性过高的缺点，这可能会影响最终结果。

2）深度森林回归模型虽然在精度和训练速度方面都有优势，但其持久化后需要的内存远大于其他模型。

3）原型网络虽然利用了神经网络去处理小样本的分类问题，但由于我们使用的 137 维特征作为输入，所以训练出来的模型仍然可能面临过拟合的风险。

4）使用粒子群算法的确达到了部分优化效果，但在原始数据中存在虽然 ADMET 性质仅部分优秀，但生物活性达到 9.860121 的化合物。这说明模型并未达到最优，粒子群算法可能陷入了局部最优。

9.3 模型的改进与推广

1) 在问题一进行变量筛选的过程中，我们采用了异质集成的方式。我们还可以尝试同质集成的方式，采用 bootstrap 方式选择不同的数据集，使用同种算法评估重要性并进行集成，在集成方式上也可选择例如 svm-rank 等方式进行排序。

2) 在生物活性预测中，虽然 Stacking 模型的效果不如深度森林模型，但这可能和基学习器的选取有关，如何选择更好的基学习器以及如何优化参数仍有可能得到更好的结果。

3) 针对基于原型网络的 ADMET 二分类模型，虽然我们评估了变量的重要性程度，但并未进一步深究，我们可以利用递归特征消除的方式，通过逐步删除特征的方式训练更鲁棒的原型网络分类模型，从而降低过拟合风险。

4) 针对第四问的优化模型，其结果与前两问的模型息息相关，由于我们在 ADMET 分类中使用了 137 个生物描述符，其过高的维度加大了优化的难度，我们可以通过使用新的优化算法或改进分类模型的方式进行提升。

5) 针对四个问题的维恩图分析，本文得出了显著影响化合物活性和 ADMET 性质的三个分子描述符 **MDEC-23、SaasC 和 C3SP2**，这对治疗乳腺癌药物选取具有一定的指导意义。

参考文献

[1] Tang Y, Wang Y, Kiani MF, Wang B. Classification, Treatment Strategy, and Associated Drug Resistance in Breast Cancer. Clin Breast Cancer. 2016,16(5):335-343.

[2] Zhang Y, Wei S, Dong C, et al. 定量结构-性质关系 (QSPR) 中的计算方法研究进展[J]. Chinese Science Bulletin. 2021

[3] Study of Peak Load Demand Estimation Methodology by Pearson Correlation Analysis with Macro-economic Indices and Power Generation Considering Power Supply Interruption. Journal of Electrical Engineering & Technology, 2017,12(4).

[4] 胡军，张超，陈平雁，非参数双变量相关分析方法 Spearman 和 Kendall 的 Monte Carlo 模拟比较[J]. 中国卫生统计,2008,25(06):590-591.

[5] 徐维超. 相关系数研究综述[J]. 广东工业大学学报, 2012(03):12-17.

[6] Remeseiro B , Bolon-Canedo V . A review of feature selection methods in medical applications[J]. Computers in Biology and Medicine. 2019:103375.

[7] Székely G J, Rizzo M L, Bakirov N K. Measuring and testing dependence by correlation of distances[C]. Acm Symposium on Virtual Reality Software & Technology, 2007.

[8] Breiman L . Using Iterated Bagging to Debias Regressions[J]. Machine Learning, 2001, 45(3):261-277.

[9] Zhou Z H , Feng J . Deep Forest: Towards An Alternative to Deep Neural Networks[J]. 2017.

[10] Snell J , Swersky K , Zemel R S . Prototypical Networks for Few-shot Learning[J]. 2017.

[11] Dana-Song,分类模型的评价指标--混淆矩阵: https://blog.csdn.net/shy19890510/article/details/79501582，2020-09-20.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

附录

问题一代码

```
import pandas as pd
import numpy as np
from scipy.spatial.distance import pdist, squareform
from sklearn.linear_model import ElasticNetCV,LassoCV,Lasso,ElasticNet
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import VarianceThreshold
from sklearn.model_selection import cross_validate,train_test_split
from ITMO_FS.filters.univariate import f_ratio_measure,pearson_corr,spearman_corr,kendall_corr
from sklearn.inspection import permutation_importance
from sklearn.preprocessing import RobustScaler,StandardScaler,PowerTransformer
from sklearn.cross_decomposition import PLSRegression
from sklearn.metrics import r2_score

from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor,GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error,r2_score
from sklearn.linear_model import ElasticNet,Lasso
from deepforest import CascadeForestRegressor

import seaborn as sns
from sklearn.svm import SVR
import warnings
warnings.filterwarnings('ignore')

def constant_feature_detect(data,threshold=0.98):
        data_copy = data.copy(deep=True)
    quasi_constant_feature = []
    for feature in data_copy.columns:
        predominant = (data_copy[feature].value_counts() / np.float(
                            len(data_copy))).sort_values(ascending=False).values[0]
        if predominant >= threshold:
            quasi_constant_feature.append(feature)
    print(len(quasi_constant_feature),' variables are found to be almost constant')
    print(quasi_constant_feature)
```

```
    return quasi_constant_feature

def distcorr(X, Y):
    X = np.atleast_1d(X)
    Y = np.atleast_1d(Y)
    if np.prod(X.shape) == len(X):
        X = X[:, None]
    if np.prod(Y.shape) == len(Y):
        Y = Y[:, None]
    X = np.atleast_2d(X)
    Y = np.atleast_2d(Y)
    n = X.shape[0]
    if Y.shape[0] != X.shape[0]:
        raise ValueError('Number of samples must match')
    a = squareform(pdist(X))
    b = squareform(pdist(Y))
    A = a - a.mean(axis=0)[None, :] - a.mean(axis=1)[:, None] + a.mean()
    B = b - b.mean(axis=0)[None, :] - b.mean(axis=1)[:, None] + b.mean()

    dcov2_xy = (A * B).sum()/float(n * n)
    dcov2_xx = (A * A).sum()/float(n * n)
    dcov2_yy = (B * B).sum()/float(n * n)
    dcor = np.sqrt(dcov2_xy)/np.sqrt(np.sqrt(dcov2_xx) * np.sqrt(dcov2_yy))
    return dcor

molecular_des = pd.read_excel('../../data/Molecular_Descriptor.xlsx',engine='openpyxl',index_col=0,)
era_activity = pd.read_excel('../../data/ERα_activity.xlsx',engine='openpyxl',index_col=0,)
quasi_constant_feature=constant_feature_detect(data=molecular_des,threshold=0.9)
molecular_des.drop(labels=quasi_constant_feature,axis=1,inplace=True)
molecular_des = molecular_des.loc[:,molecular_des.std() >= 0.05]
all_data = molecular_des.join(era_activity)
X = all_data.iloc[:,:-2].values
y = all_data.iloc[:,-1].values
rs = StandardScaler()
X = rs.fit_transform(X)
pearson_score = pearson_corr(X,y)
pearson_importance = np.abs(pearson_score) / np.sum(np.abs(pearson_score))
pearson_rank = np.argsort(-1 * pearson_importance)
distcorr_list = []
for i in range(X.shape[1]):
    distcorr_list.append(distcorr(X[:,i],y))
distcorr_importance = np.abs(distcorr_list) / np.sum(np.abs(distcorr_list))
dist_corr_rank = np.argsort(-1 * distcorr_importance)
rfr = RandomForestRegressor(n_jobs=-1)
```

```
rfr.fit(X,y)
rf_importance = rfr.feature_importances_
rf_rank = np.argsort(-1 * rfr.feature_importances_)
en = ElasticNet(l1_ratio=0.00)
en.fit(X,y)
en_importance = np.abs(en.coef_) / np.sum(np.abs(en.coef_))
en_rank = np.argsort(-1 * abs(en.coef_))
total_importance =   0.1 * pearson_importance +   0.1 * distcorr_importance +   0.5 * rf_importance +   0.3
* en_importance
total_importance = total_importance / total_importance.sum()
feature_rank = np.argsort(-1 * total_importance)
###划分数据集###
X_train_df, X_test_df= train_test_split(molecular_des,test_size=0.3, random_state=56)
train_df = X_train_df.join(era_activity)
test_df = X_test_df.join(era_activity)
X_train = train_df.iloc[:,:-2].values
y_train = train_df.iloc[:,-1].values
X_test = test_df.iloc[:,:-2].values
y_test = test_df.iloc[:,-1].values

rs = StandardScaler()
rs.fit(X_train)
X_train = rs.transform(X_train)
X_test = rs.transform(X_test)

X_train_filter = X_train[:,feature_rank[:20]]
X_test_filter = X_test[:,feature_rank[:20]]

names = ["KNeighborsRegressor", "RBF SVR",
            "RandomForestRegressor","GradientBoostingRegressor",
"AdaBoostRegressor","CascadeForestRegressor","ElasticNet"]

classifiers = [
      KNeighborsRegressor(),
      SVR(kernel="rbf"),
      RandomForestRegressor(),
      GradientBoostingRegressor(n_estimators=200),
      AdaBoostRegressor(),
      CascadeForestRegressor(),
      ElasticNet()
      ]

regressor_df = pd.DataFrame(index = names,columns=['MSE','R2'])
```

```
for name, clf in zip(names, classifiers):
    print("---start\t" + name + "\t----")
    clf.fit(X_train_filter, y_train)
    y_pred = clf.predict(X_test_filter)
    MSE = mean_absolute_error(y_pred,y_test)
    R2 = r2_score(y_test,y_pred)
    regressor_df.loc[name]['MSE']=MSE
    regressor_df.loc[name]['R2'] = R2
```

问题二代码

```
X = all_data.iloc[:,:-2].values
y = all_data.iloc[:,-1].values
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.3, random_state=56)
######  证明 stack 算法优于单个算法

X = all_data.iloc[:,:-2].values
y = all_data.iloc[:,-1].values
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.3, random_state=56)

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_valid = scaler.transform(X_valid)

names = ["KNeighborsRegressor", "RBF SVR",
         "RandomForestRegressor",
         "GradientBoostingRegressor",
         "Stacking",
         "CascadeForestRegressor"]

estimators = [('rf', RandomForestRegressor(n_jobs=-1)),('svr',
SVR(kernel="rbf")),('gdbt',GradientBoostingRegressor()),
              ('knn',KNeighborsRegressor(n_jobs=-1))]

classifiers = [
    KNeighborsRegressor(n_jobs=-1),
    SVR(kernel="rbf"),
    RandomForestRegressor(n_jobs=-1),
    GradientBoostingRegressor(),
    StackingRegressor(estimators=estimators,final_estimator=LinearRegression()),
    CascadeForestRegressor(verbose=False,n_jobs=-1,random_state=56),
    ]
```

```
regressor_df = pd.DataFrame(index = names,columns=['MSE','R2'])
predict_map = {}
for name, clf in zip(names, classifiers):
    print("---start\t" + name + "\t----")
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_valid)
    predict_map[name] = y_pred
    MSE = mean_absolute_error(y_pred,y_valid)
    R2 = r2_score(y_valid,y_pred)
    regressor_df.loc[name]['MSE']=MSE
    regressor_df.loc[name]['R2'] = R2


#### 参数优化
import optuna
from optuna.samplers import TPESampler

def deepforest_objective(trial):

    df_max_layers = trial.suggest_int('df_max_layers', 2, 5)
    df_n_trees = trial.suggest_int('df_n_trees', 100, 500,50)
    df_min_samples_leaf = trial.suggest_int('df_min_samples_leaf', 1, 20,5)

    X = all_data.iloc[:,:-2].values
    y = all_data.iloc[:,-1].values
    X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.3, random_state=56)

    scaler = StandardScaler()
    deepforest = CascadeForestRegressor(max_layers=df_max_layers,min_samples_leaf =
df_min_samples_leaf,n_trees=df_n_trees,verbose=True,n_jobs=-1,
                                        random_state=56)
    stackingPipe = Pipeline([('scaler', StandardScaler()), ('deepforest', deepforest)])
    stackingPipe.fit(X_train,y_train)
    y_pred = stackingPipe.predict(X_valid)
    return mean_absolute_error(y_pred,y_valid)

study = optuna.create_study(direction='minimize',sampler=TPESampler())
study.optimize(deepforest_objective,n_trials=50)
print(study.best_params)
print(study.best_value)
```

问题三代码

```
import torch
import torch.nn as nn
import random

import torchvision
from scipy.sparse import csr_matrix
from torch import optim
import torch.nn.functional as F
from torch.autograd import Variable
from torch.utils.data import Dataset, DataLoader
import matplotlib.pyplot as plt

torch.manual_seed(999)
np.random.seed(999)
random.seed(999)

class EarlyStopping:
    """Early stops the training if validation loss doesn't improve after a given patience."""
    def __init__(self, patience=7, verbose=False, delta=0, path='finish_model.pkl', trace_func=print):
        """
        Args:
            patience (int): How long to wait after last time validation loss improved.
                            Default: 7
            verbose (bool): If True, prints a message for each validation loss improvement.
                            Default: False
            delta (float): Minimum change in the monitored quantity to qualify as an improvement.
                            Default: 0
            path (str): Path for the checkpoint to be saved to.
                            Default: 'checkpoint.pt'
            trace_func (function): trace print function.
                            Default: print
        """
        self.patience = patience
        self.verbose = verbose
        self.counter = 0
        self.best_score = None
        self.early_stop = False
        self.val_loss_min = np.Inf
        self.delta = delta
        self.path = path
        self.trace_func = trace_func
    def __call__(self, val_loss, model):
        score = -val_loss
```

```
            if self.best_score is None:
                self.best_score = score
                self.save_checkpoint(val_loss, model)
            elif score < self.best_score + self.delta:
                self.counter += 1
                self.trace_func(f'EarlyStopping counter: {self.counter} out of {self.patience}')
                if self.counter >= self.patience:
                    self.early_stop = True
            else:
                self.best_score = score
                self.save_checkpoint(val_loss, model)
                self.counter = 0

        def save_checkpoint(self, val_loss, model):
            '''Saves model when validation loss decrease.'''
            if self.verbose:
                self.trace_func(f'Validation loss decreased ({self.val_loss_min:.6f} --> {val_loss:.6f}).  Saving
model ...')
            # torch.save(model.state_dict(), self.path)
            torch.save(model, self.path)
            self.val_loss_min = val_loss

class MulitiPrototypicalNet3(nn.Module):
        def __init__(self,in_feature,num_class, embedding_dim,
                     support_ratio = 0.6,query_ratio = 0.3,hidden1_dim = 1024,hidden2_dim =
256,distance='euclidean'):
            super(MulitiPrototypicalNet3, self).__init__()
            self.num_class = num_class
            self.embedding_dim = embedding_dim
            self.support_ratio = support_ratio
            self.query_ratio = query_ratio
            self.support_num = []
            self.query_num = []
            self.distance = distance
            self.prototype = None
            self.prototypes = []

            self.feature_extraction = nn.Sequential(
                nn.Linear(in_features=in_feature, out_features=hidden1_dim),
                nn.ReLU(inplace=True),
                nn.Dropout(p=0.5),
                nn.Linear(in_features=hidden1_dim, out_features=hidden2_dim),
                nn.ReLU(inplace=True),
                nn.Dropout(p=0.5),
```

```
                nn.Linear(in_features=hidden2_dim, out_features=embedding_dim),
            )
        def weights_init_(self):
            for m in self.modules():
                torch.nn.init.xavier_normal_(m.weight, gain=1, )
                torch.nn.init.constant_(m.bias, 0)

        def embedding(self, features):
            result = self.feature_extraction(features)
            return result

        def forward(self, support_input, query_input):
            return result

        def randomGenerate(self, X, Y):
            return support_input, query_input, support_label, query_label

        def fit(self,X_train,y_train,X_valid,y_valid,optimizer,criterion,patience,EPOCH):
            return loss_list

        def predict(self,X_test):
            return pre_Y, prob_Y

def miv(model, X):
        model.eval()
        miv = torch.ones(X.shape[1])
        for i in range(X.shape[1]):
            cur_X_1 = X.copy()
            cur_X_2 = X.copy()
            cur_X_1[:, i] = cur_X_1[:, i] + cur_X_1[:, i] * 0.1
            cur_X_2[:, i] = cur_X_2[:, i] - cur_X_2[:, i] * 0.1
            cur_X_1 = torch.tensor(cur_X_1, dtype=torch.float)
            cur_X_2 = torch.tensor(cur_X_2, dtype=torch.float)

            cur_diff = torch.mean(model.embedding(cur_X_1) - model.embedding(cur_X_2), dim=1)
            miv[i] = torch.mean(cur_diff, dim=0)
        s = torch.abs(miv) / torch.sum(torch.abs(miv))
        rank = torch.argsort(torch.abs(miv), dim=0, descending=True)
        return rank, s

molecular_des_corr_del = pd.read_csv("molecular_des_corr_del.csv",index_col=0)
all_data = molecular_des_corr_del.join(admet)
X = all_data.iloc[:,:-5].values
y = all_data.iloc[:,-5].values
```

```
X_train, X_valid, y_train, y_valid = train_test_split( X, y, test_size=0.3, stratify=y,random_state=56)
ss = StandardScaler()
ss.fit(X_train)
X_train = ss.transform(X_train)
X_valid = ss.transform(X_valid)
joblib.dump(ss, 'p1_ss.pkl')
model = MulitiPrototypicalNet3(X_train.shape[1],2, 24,)
optimizer = optim.Adam(model.parameters(), lr=0.001, weight_decay=0.001)
criterion = nn.CrossEntropyLoss()
model.fit(X_train,y_train,X_valid,y_valid,optimizer,criterion,50,500)
pre_Y, prob_Y = model.predict(X_valid)
final_model1 = torch.load("./finish_model_1.pkl")
pre_valid_y_model1, prob_valid_y_model1 = final_model1.predict(X_valid)
acc = accuracy_score(y_valid, pre_valid_y_model1)
auc = roc_auc_score(y_valid,prob_valid_y_model1)
print('acc:     ' + str(acc))
print('auc:     ' + str(auc))
rank,s = miv(final_model1,X_train)
show_feature_num = 10

f_importance = pd.DataFrame({"feature": molecular_des_corr_del.columns,"importance":
s.detach().numpy()})
f_importance = f_importance.sort_values(by="importance", ascending=False)
# print(f_importance)
f_importance = f_importance[:show_feature_num]
p = sns.barplot(x="importance", y="feature", data=f_importance,
                order=f_importance["feature"],orient="h",palette=sns.color_palette("tab10",
show_feature_num),)
p.set_title('Caco-2 Feature importance')
bar_fig = p.get_figure()
plt.show()
```

问题 4 代码

```
import numpy as np
from functools import lru_cache
from types import MethodType, FunctionType
import warnings
import sys
def func_transformer(func):

    if (func.__class__ is FunctionType) and (func.__code__.co_argcount > 1):
        warnings.warn('multi-input might be deprecated in the future, use fun(p) instead')
```

```
        def func_transformed(X):
            return np.array([func(*tuple(x)) for x in X])

        return func_transformed

    if (func.__class__ is MethodType) and (func.__code__.co_argcount > 2):
        warnings.warn('multi-input might be deprecated in the future, use fun(p) instead')

        def func_transformed(X):
            return np.array([func(tuple(x)) for x in X])

        return func_transformed

    if getattr(func, 'is_vector', False):
        warnings.warn('''
        func.is_vector will be deprecated in the future, use set_run_mode(func, 'vectorization') instead
        ''')
        set_run_mode(func, 'vectorization')

    mode = getattr(func, 'mode', 'others')
    valid_mode = ('common', 'multithreading', 'multiprocessing', 'vectorization', 'cached', 'others')
    assert mode in valid_mode, 'valid mode should be in ' + str(valid_mode)
    if mode == 'vectorization':
        return func
    elif mode == 'cached':
        @lru_cache(maxsize=None)
        def func_cached(x):
            return func(x)

        def func_warped(X):
            return np.array([func_cached(tuple(x)) for x in X])

        return func_warped
    elif mode == 'multithreading':
        from multiprocessing.dummy import Pool as ThreadPool

        pool = ThreadPool()

        def func_transformed(X):
            return np.array(pool.map(func, X))

        return func_transformed
    elif mode == 'multiprocessing':
        from multiprocessing import Pool
```

```
            pool = Pool()

            def func_transformed(X):
                return np.array(pool.map(func, X))

            return func_transformed

        else:  # common
            def func_transformed(X):
                return np.array([func(x) for x in X])

            return func_transformed

from abc import ABCMeta, abstractmethod
import types
import warnings

class SkoBase(metaclass=ABCMeta):
    def register(self, operator_name, operator, *args, **kwargs):
        def operator_wapper(*wrapper_args):
            return operator(*(wrapper_args + args), **kwargs)

        setattr(self, operator_name, types.MethodType(operator_wapper, self))
        return self

    def fit(self, *args, **kwargs):
        warnings.warn('.fit() will be deprecated in the future. use .run() instead.'
                      , DeprecationWarning)
        return self.run(*args, **kwargs)
class Problem(object):
    pass
class PSO(SkoBase):
    def __init__(self,chushi_pop ,func, n_dim=None, pop=40, max_iter=150, lb=-1e5, ub=1e5, w=0.8,
c1=0.5, c2=0.5,
                 constraint_eq=tuple(), constraint_ueq=tuple(), verbose=False
                 , dim=None):

        n_dim = n_dim or dim  # support the earlier version

        self.func = func_transformer(func)
        self.w = w  # inertia
        self.cp, self.cg = c1, c2  # parameters to control personal best, global best respectively
        self.pop = pop  # number of particles
        self.n_dim = n_dim  # dimension of particles, which is the number of variables of func
```

```
        self.max_iter = max_iter    # max iter
        self.verbose = verbose    # print the result of each iter or not
        self.lb, self.ub = np.array(lb) * np.ones(self.n_dim), np.array(ub) * np.ones(self.n_dim)
        assert self.n_dim == len(self.lb) == len(self.ub), 'dim == len(lb) == len(ub) is not True'
        assert np.all(self.ub > self.lb), 'upper-bound must be greater than lower-bound'
        self.has_constraint = bool(constraint_ueq)
        self.constraint_ueq = constraint_ueq
        self.is_feasible = np.array([True] * pop)

        self.X = np.vstack((chushi_pop,np.random.uniform(low=self.lb, high=self.ub, size=(self.pop - chushi_np.shape[0], self.n_dim))))

        print(self.X.shape[1])
        for i in range(self.X.shape[1]):
            if data_type[:,i]== 1:
                self.X[:,i] = self.X[:,i].astype(np.int64)
        print(self.X.shape)
        v_high = self.ub - self.lb
        self.V = np.random.uniform(low=-v_high, high=v_high, size=(self.pop, self.n_dim))  # speed of particles
        self.Y = self.cal_y()    # y = f(x) for all particles
        self.pbest_x = self.X.copy()    # personal best location of every particle in history
        self.pbest_y = np.array([[np.inf]] * pop)    # best image of every particle in history
        self.gbest_x = self.pbest_x.mean(axis=0).reshape(1, -1)    # global best location for all particles
        self.gbest_y = np.inf    # global best y for all particles
        self.gbest_y_hist = []    # gbest_y of every iteration
        self.update_gbest()

        # record verbose values
        self.record_mode = True
        self.record_value = {'X': [], 'V': [], 'Y': []}
        self.all_best_x = []
        self.all_best_y = []
        self.best_x, self.best_y = self.gbest_x, self.gbest_y    # history reasons, will be deprecated

    def check_constraint(self, x):


        if self.constraint_ueq(x) > 0:
            return False
        return True

    def update_V(self):
        r1 = np.random.rand(self.pop, self.n_dim)
```

```
        r2 = np.random.rand(self.pop, self.n_dim)
        self.V = self.w * self.V + \
                    self.cp * r1 * (self.pbest_x - self.X) + \
                    self.cg * r2 * (self.gbest_x - self.X)
        for i in range(self.V.shape[1]):
            if data_type[:,i]== 1:
                self.V[:,i] = self.V[:,i].astype(np.int64)

    def update_X(self):
        self.X = self.X + self.V
        self.X = np.clip(self.X, self.lb, self.ub)

    def cal_y(self):
        # calculate y for every x in X
        self.Y = self.func(self.X).reshape(-1, 1)
        return self.Y
    def update_pbest(self):
        '''
        personal best
        :return:
        '''
        self.need_update = self.pbest_y > self.Y
        for idx, x in enumerate(self.X):
            if self.need_update[idx]:
                self.need_update[idx] = self.check_constraint(x)
        self.pbest_x = np.where(self.need_update, self.X, self.pbest_x)
        self.pbest_y = np.where(self.need_update, self.Y, self.pbest_y)
    def update_gbest(self):
        '''
        global best
        :return:
        '''
        idx_min = self.pbest_y.argmin()
        if self.gbest_y > self.pbest_y[idx_min]:
            self.gbest_x = self.X[idx_min, :].copy()
            self.gbest_y = self.pbest_y[idx_min]

    def recorder(self):
        if not self.record_mode:
            return
        self.record_value['X'].append(self.X)
        self.record_value['V'].append(self.V)
        self.record_value['Y'].append(self.Y)
```

```
    def run(self, max_iter=None, precision=1e-7, N=20):
        '''
        precision: None or float
            If precision is None, it will run the number of max_iter steps
            If precision is a float, the loop will stop if continuous N difference between pbest less than precision
        N: int
        '''
        self.max_iter = max_iter or self.max_iter
        c = 0
        for iter_num in range(self.max_iter):
            print("当前迭代次数:\t" + str(iter_num))
            self.update_V()
            self.recorder()
            self.update_X()
            self.cal_y()
            self.update_pbest()
            self.update_gbest()
            if precision is not None:
                tor_iter = np.amax(self.pbest_y) - np.amin(self.pbest_y)
                if tor_iter < precision:
                    c = c + 1
                    if c > N:
                        break
                else:
                    c = 0
            if self.verbose:
                print('Iter: {}, Best fit: {} at {}'.format(iter_num, self.gbest_y, self.gbest_x))

            self.gbest_y_hist.append(self.gbest_y)
        self.best_x, self.best_y = self.gbest_x, self.gbest_y
        return self.best_x, self.best_y
molecular_des_corr_del = pd.read_csv('../molecular_des_corr_del.csv',index_col=0)
feature_20 = pd.read_csv('../20feature.csv',index_col=0)
feature_20_index = []
for i in feature_20.columns:
    feature_20_index.append(molecular_des_corr_del.columns.get_loc(i))
molecular_min = molecular_des_corr_del_describe.loc['min',:]
molecular_max = molecular_des_corr_del_describe.loc['max',:]
admet = pd.read_excel('../../data/ADMET.xlsx',sheet_name="training",engine='openpyxl',index_col=0,)
admet['hERG'] = 1 - admet['hERG']
admet['MN'] = 1 - admet['MN']
good_admet = admet[admet.sum(axis=1) >= 3]
era_activity = pd.read_excel('../../data/ERα_activity.xlsx',engine='openpyxl',index_col=0,)
```

```
join_data = good_admet.join(era_activity)
sifenweishu = join_data['pIC50'].quantile([0.75]).values[0]
good_sample_name = join_data[join_data['pIC50'] > 8].index.tolist()
chushi_np = molecular_des_corr_del.loc[good_sample_name,:].values
import pickle
import joblib
import torch
pkl_filename = "../问题 2/deepforest_model.pkl"
with open(pkl_filename, 'rb') as file:
    load_deepforest = pickle.load(file)

p2_ss = joblib.load("../问题 3/p2_ss.pkl")
pn_model1 = torch.load("../问题 3/finish_model_1.pkl")
pn_model2 = torch.load("../问题 3/finish_model_2.pkl")
pn_model3 = torch.load("../问题 3/finish_model_3.pkl")
pn_model4 = torch.load("../问题 3/finish_model_4.pkl")
pn_model5 = torch.load("../问题 3/finish_model_5.pkl")
molecular_des_corr_del.info()
data_type = np.zeros((1,molecular_des_corr_del.shape[1]))
for i in range(0,molecular_des_corr_del.shape[1]):
    if molecular_des_corr_del.iloc[:,i].dtype == 'int64':
        data_type[:,i]=1
data_type
def demo_func(x):
    x = x.reshape(-1,1)
    x1 = x[feature_20_index].reshape(1,-1)
    x = x.reshape(1,-1)
    x = p2_ss.transform(x)
    return -(0.7*load_deepforest.predict(x1)
                    +0.3*( pn_model1.predict(x)[0]
                    +pn_model2.predict(x)[0]
                    +(1-pn_model3.predict(x)[0])
                    +pn_model4.predict(x)[0]
                    +(1-pn_model5.predict(x)[0])))

constraint_ueq = (
    lambda x:   3 - (pn_model1.predict(p2_ss.transform(x.reshape(1,-1)))[0]
                    +pn_model2.predict(p2_ss.transform(x.reshape(1,-1)))[0]
                    +(1-pn_model3.predict(p2_ss.transform(x.reshape(1,-1)))[0])
                    +pn_model4.predict(p2_ss.transform(x.reshape(1,-1)))[0]
                    +(1-pn_model5.predict(p2_ss.transform(x.reshape(1,-1)))[0]))

)
pso   =   PSO(chushi_pop=chushi_np,func=demo_func,constraint_ueq=constraint_ueq,n_dim=137,   pop=40,
```

```
max_iter=100, lb=molecular_min.values, ub=molecular_max.values, w=0.8, c1=0.5, c2=0.5)
pso.run()
print('best_x is ', pso.gbest_x, 'best_y is', pso.gbest_y)
```