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		山西大学
参赛队号		**21101080006**
队员姓名	**1.**	祝海峰
	2.	陈庆辉
	3.	孟颖岫

中国研究生创新实践系列大赛

“华为杯”第十八届中国研究生数学建模竞赛

题 目 抗乳腺癌候选药物的优化建模

摘 要：

乳腺癌是目前世界上最常见，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，能够拮抗 ERα活性的化合物可能是治疗乳腺癌的候选药物。本文在保证了 ADMET 性质优越的情况下建立了化合物活性预测模型，来确定影响化合物活性的主要分子描述符及其取值范围，为生产中抗乳腺癌化合物的选择提供理论依据。

针对第一问，筛选出影响生物活性最显著的 20 个分子描述符。首先，本文对数据进行了预处理：将含 0 比例大于 90%的特征剔除，剔除了 344 个特征；用拉依达准则对剩余数据进行异常检测，剔除异常值数量在 100 以上的特征，有 26 个特征被剔除；对于异常值数量在 100 以内的特征，将异常值进行限幅。然后，把 pIC50 作为因变量，用随机森林和熵值法对剩余 359 个特征进行分析，分别筛选出 30 个主要变量，并分别画出二者 30 个变量的相关性图，对比发现随机森林筛选的特征更具有代表性。接着，对随机森林筛选的 30 个变量进行 Pearson 相关性分析，将强关系的变量进行逐一剔除，最终获得影响生物活性的 20 个主要变量。

针对第二问，构建化合物对 ERα生物活性的定量回归预测模型。首先，以 pIC50 为因变量，第一问筛选的 20 个变量作为自变量，建立支持向量机、神经网络、梯度提升和随机森林的回归预测模型，并用随机搜索方法搜索四种模型的最佳超参数。然后，用 MAE、RMSE 和拟合度 3 个评价指标对四种模型进行评价，并画出四种模型对测试集前 20 组数据实际值和预测值的拟合图，观察发现随机森林效果最佳。最后，用随机森林预测题目中给的 50 组化合物的 pIC50，并通过公式求解出 IC50_nM。

针对第三问，分别构建五种化合物的分类预测模型。首先，使用第一问中数据预处理之后剩余的 359 个特征数据作为自变量，五个化合物的二分类数据作为因变量。然后，构建五种化合物的四个二分类模型：支持向量机、随机梯度下降、神经网络和随机森林。以 ROC、AUC 和准确率作为模型评价标准，用随机搜索对五个化合物分类模型分别进行最佳超参数搜索，最终发现随机森林对五种化合物的分类效果均优于其他模型。确定使用随机森林作为五种化合物的分类预测模型，并对题中给的 50 组数据进行分类预测。

对于第四问，建立化合物活性优化模型，来选取适合的分子描述符及其范围。使用第二问 pIC50 回归模型和第三问的 ADMET 分类模型对此问进行求解。以 pIC50 为优化目标，ADMET 最少三个性质较好为约束条件，pIC50 回归模型中的 20 个变量作为决策变量，建立目标规划模型。然后，使用遗传算法对该目标规划模型进行求解。在求解过程中，画出来了优化迭代曲线，并对比了优化后和优化前的 pIC50 分布，可以清晰地看出优化后 pIC50 平均值明显增大，波动变小。由此可得出 20 个决策变量对优化模型有较强的优化效果，

把这 20 个变量作为要寻找的分子描述符变量，并取它们的最大-最小值作为优化取值范围。最后，对 20 个分子描述符进行敏感性分析，采取控制变量法观察自变量在-50%-50%范围内以 1%步长浮动时对因变量的影响，来进一步修正 20 个分子描述符的取值范围。

关键词：化合物活性优化模型、支持向量机、神经网络、梯度提升、随机森林的回归预测、随机梯度下降、目标规划、遗传算法

目录

1、问题背景与问题重述

1.1 问题背景

癌症是威胁人类健康的重大疾病，据世界卫生组织（WHO） 癌症研究机构（IARC）最新发布的《全球癌症报告》显示， 2020 年全球新发癌症病例 1929 万，由癌症导致的死亡病例 996 万。乳腺癌是目前世界上最常见、致死率较高的癌症之一，一旦发现后就应该马上治疗，坚持治疗肿瘤不可或缺。雌激素受体α亚型（Estrogen receptors alpha, ERα）被认为是治疗乳腺癌的重要靶标，能够拮抗 ERα活性的化合物被认为可能是治疗乳腺癌的候选药物。在治疗乳腺癌药物的研发过程中，为了节约时间和成本，通常采用建立化合物活性预测模型的方法来筛选潜在活性化合物。通过收集到的一系列作用于靶标 ERα的化合物和其生物活性数据，构建以化合物的分子结构描述符为自变量，化合物的生物活性值为因变量的定量结构-活性关系模型，使用该模型不仅可以指导目前已知的生物活性化合物的结构优化，而且可以预测具有更好生物活性的新化合物分子，使其具有更好的生物活性，为乳腺癌的治疗提供新的解决方案。在药物研发领域，评判一个化合物是否能够成为候选药物，除了需要具备良好的生物活性外，还需要在人体内具有良好的药代动力学性质和安全性。即该化合物进入人体后需要具备良好的吸收和代谢性质并且不会对人体产生隐藏的毒性隐患，以上性质被合称为 ADMET 性质（Absorption 吸收、Distribution 分布、Metabolism 代谢、Excretion 排泄、Toxicity 毒性）。因此为抗乳腺癌候选药物构建一个基于生物活性和 ADMET 性质的精确、稳定、高效的化合物分子定量结构-生物活性关系模型对于乳腺癌的治疗具有很重要的实际意义。

1.2 问题重述

基于上述问题背景与提供的附件数据，本文需要研究解决以下问题：

问题一：筛选分子描述符

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，给出前 20 个对生物活性最具有显著影响的分子描述符并详细说明筛选过程及其合理性。，

问题二：构建定量预测模型

结合问题 1，选择不超过 20 个分子描述符变量，构建化合物对 ERα生物活性的定量预测模型并叙述建模过程。然后使用构建的预测模型，对文件“ERα_activity.xlsx”的 test 表中的 50 个化合物进行 IC50 和 pIC50 值的预测。

问题三：构建分类预测模型

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。

问题四：化合物 ADMET 性质及其对抑制 ERα生物活性的优化

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、模型假设与符号说明

2.1 模型假设

考虑到现实问题，本文做出如下假设：

(1) 假设除了附件中的变量，没有其他变量会对生物活性产生影响；
(2) 假设附件提供的数据是真实可靠的；
(3) 假设数据中个别缺失数据对结果不会产生重大影响。

2.2 符号说明

本文中用到的变量符号及其说明如下表 2-1 所示：

表 2-1 变量符号及其说明

序号	符号	符号说明
1	$\mathrm{cov}(X,Y)$	X、Y 的协方差
2	σ_X	X 的标准差
3	σ_Y	Y 的标准差
4	E	数学期望
5	MAE	平均绝对误差
6	RMSE	均方根误差
7	TSS	样本的总平方和
8	RSS	残差平方和
9	Y_actual	Y 的真实值
10	Y_predict	Y 的预测值
11	γ	SVM 高斯核函数超参数
12	$\left\|x-\ell\right\|$	向量的范数
13	α	SGD 下降系数
14	C	SVM 中惩罚系数
15	gamma	SVM 径向基函数自带的参数
16	max_depth	随机森林最大深度
17	n_estimators	随机森林中分类器的个数
18	max_features	随机森林允许单个决策树使用特征的最大数量

3、问题一的模型建立与求解

3.1 问题分析

问题一要求我们针对 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）。本文对上述数据的预处理的预处理包括如下过程：首先对采集的数据进行空缺值判断，发现没有空缺；接着，将含 0 比例大于 90%的特征剔除，剔除了 344 个特征；然后，采用拉依达准则对数据进行筛选，剔除特征异常值大于 100 个的特征，剔除 26 个特征；最后，对特征异常值个数小于 100 的数据进行限幅。

数据预处理完之后进行降维处理，本文采用随机森林和熵值法两种方法降维，根据贡献率各筛选 30 组主要变量，并对两种方法下的 30 组变量画相关图进行对比，发现随机森林提取的特征更具有代表性。然后对随机森林挑选的 30 组变量进行 Pearson 相关分析，逐一剔除相关性较强的 10 组变量，最后取前 20 个特征作为主要变量。

问题一的思路流程图如下图 3-1 所示：

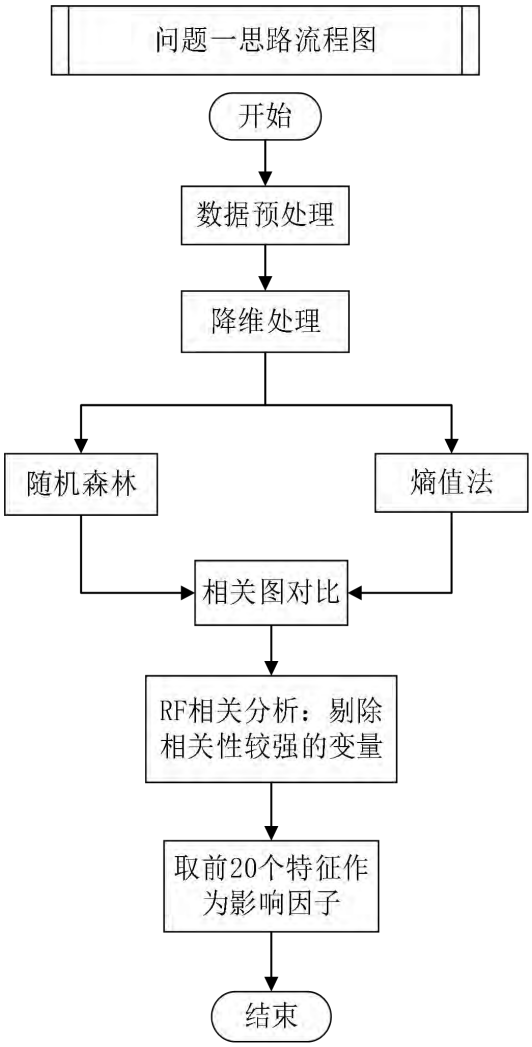


图 3-1 问题一思路流程图

3.2 数据预处理

在工程实践中，我们得到的数据会存在有缺失值、异常值等，在使用之前需要进行数据预处理。本文在所给附件的基础上做出以下处理：

首先对采集的数据进行空缺值判断，经过筛选排查没有空缺值；然后，采用拉依达准则对数据进行筛选，剔除特征值不在 $\mu \pm 3\sigma$ 范围内且异常值个数大于 100 的特征，共剔除了 26 个特征，在此基础上对特征值不在 $\mu \pm 3\sigma$ 范围内且异常值个数不大于 100 的特征进行最值限幅法处理。限幅的具体方法为：用 $\mu + 3\sigma$ 来代替大于 $\mu + 3\sigma$ 的异常值，用 $\mu - 3\sigma$ 来代替小于 $\mu - 3\sigma$ 的异常值，做到对附件中的异常数据的限幅；最后对数据进行排查筛选，剔除特征中 0 值比例大于 90%的数据，剔除 344 个特征，剩余 359 个特征，数据预处理完毕。

下图所示 3-2 为数据预处理流程图：

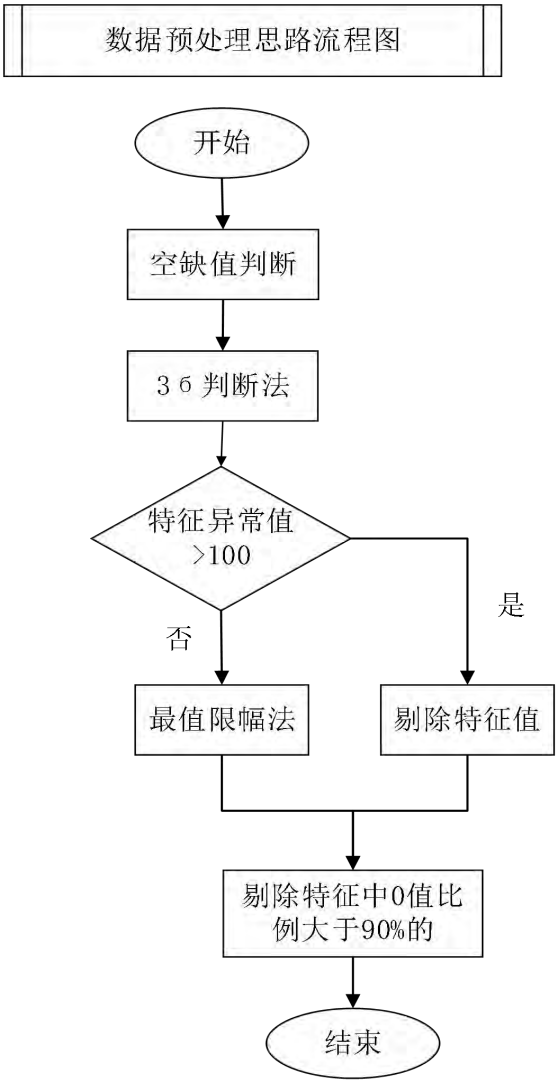


图 3-2 数据预处理流程图

本文选取了首先剔除 26 个特征中的 6 个做出其箱线图加以说明解释，如图 3-3 所示为异常数据的箱线图，其数据分布过于分散，箱子被压得很扁，甚至只剩下一条线，同时还存在很多异常值，故对此类特征进行剔除。

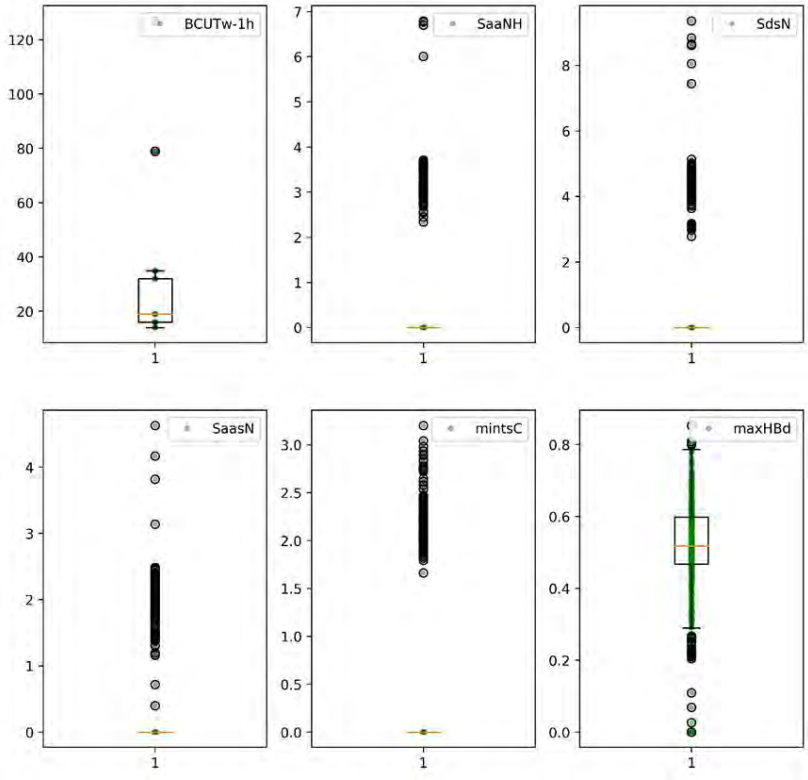


图 3-3 异常特征的箱线图

3.3 数据降维

3.3.1 皮尔逊相关分析

皮尔逊相关也称为积差相关（或积矩相关）是英国统计学家皮尔逊于 20 世纪提出的一种计算直线相关的方法。

假设有两个变量 X、Y，皮尔逊相关系数则用于度量 X 和 Y 之间的线性相关程度，其值介于-1 与 1 之间。这种线性相关直观表述为：随着 X 增大，Y 是否同时增大或者减小；当二者分布在一条直线上时，皮尔逊相关系数等于 1 或-1；两个变量之间没有线性关系时，皮尔逊相关系数为 0。

那么两变量间的皮尔逊相关系数可通过以下公式（1）计算：

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

3.3.2 熵值法降维

熵是热力学的一个物理概念，是系统混乱度（或无序度）的量度。熵越大说明系统越混乱，携带的信息越少，熵越小说明系统越有序，携带的信息越多。根据熵的特性，可以通过计算熵值来表示一个随机系统的无序程度，也可以用熵值来判断一个指标的离散程度。

熵值法的一般步骤可以表述为：数据标准化、数据预处理、计算熵值和计算权重。

我们采用熵值法降维的方法对附件“Molecular_Descriptor.xlsx”和“ERα_activity.xlsx”中的特征进行权值计算，结果如下图 3-4 所示：

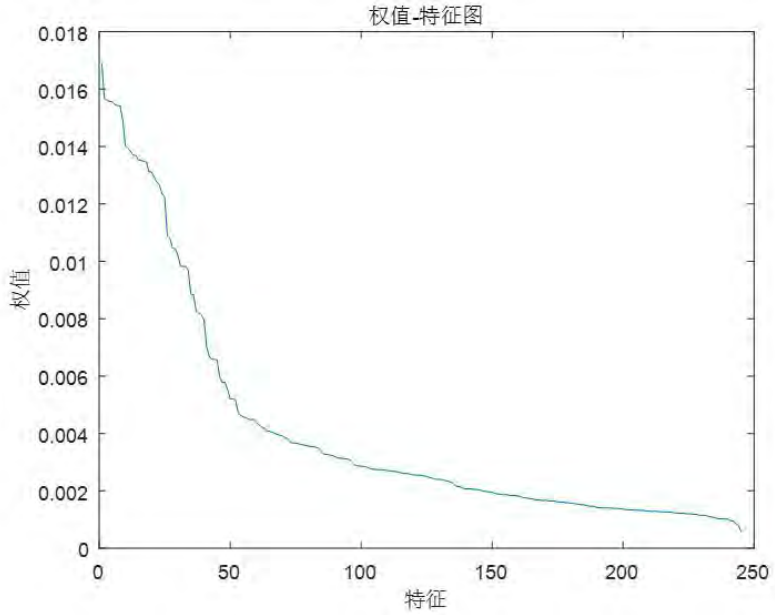


图 3-2 熵值法权值计算结果

由图示可以看出，特征值的权重逐渐下降，权值是指加权平均数中的每个数的频数，因此，在 1974 个化合物的 729 个分子描述符信息中，前 30 个特征的权值较大，说明对生物活性的影响较大，具有一定的代表性，因此，采用 python 语言对熵值法的前 30 个特征选择进行实现，结果如下图 3-4 所示：

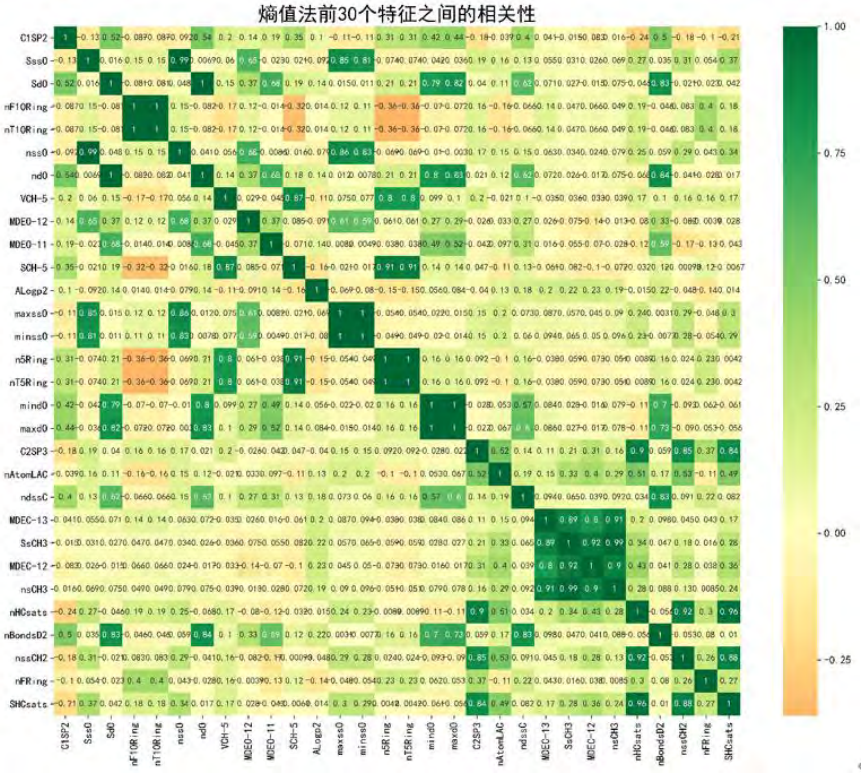


图 3-3 熵值法前 30 个特征之间的相关性

如上图 3-5 所示，在熵值法降维中，前 30 个对生物活性具有显著影响的分子描述符（即变量）之间的正相关性较强，普遍呈现绿色（绿色表示正相关性，本身之间的相关性为 1），负相关性的表示比较弱，代表性不是很强。另外，熵值法降维缺乏各对特征之间的横向比较，且各特征的权数随样本的变化而变化，权数依赖于样本，在应用上受到一定的限制。因此采用随机森林降维法弥补该缺点。

3.3.3 RF 降维法

随机森林（Random Forest，简称 RF），是一种新兴起的、高度灵活的、基于树的机器学习算法，该算法利用多棵树的力量来进行决策。森林中的每棵树并不一样，每棵树都是被随机创造的，每棵树中的每一个节点都是待选特征的一个随机子集，所有树的输出结果整合起来就是森林最后的输出结果。

随机森林重点是“随机”两个字，它包括两个方面：

训练数据的随机选取。即用于训练单棵树的数据应该是随机、有放回的从全部数据中选取和原始数据集相同的数据量。此要求是为了防止每棵树用于训练的数据一致导致训练出来的每棵树都一样，这样就丢失了构建多棵树的意义。

待选特征的随机选取。使得森林中的各棵树都能够彼此不同，提示系统的多样性，从而提升分类性能。

采用随机森林降维法对附件“Molecular_Descriptor.xlsx”和“ERα_activity.xlsx”中的特征进行处理，并根据变量对生物活性影响的重要性（贡献率）进行排序，排序后每个特征贡献率所占百分比如下图 3-6 所示：

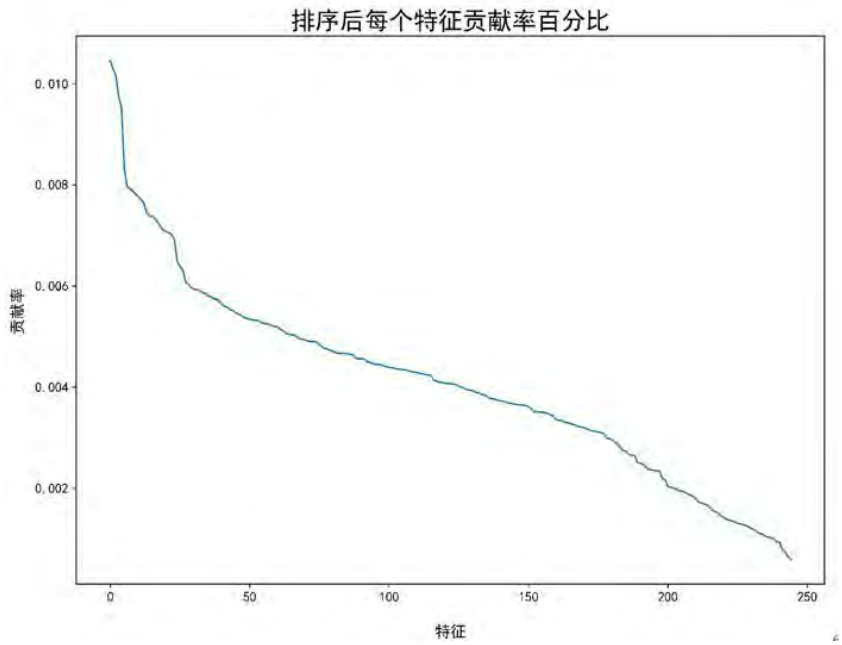


图 3-4 排序后特征贡献率所占百分比

如图 3-6 所示，对附件“Molecular_Descriptor.xlsx”和“ERα_activity.xlsx”所给的特征，进行特征贡献率计算并做好排序，在 1974 个化合物的 729 个分子描述符信息中，前 30 个特征对生物活性的贡献率较大，后面的特征贡献率小于 0.006，影响因子较小。

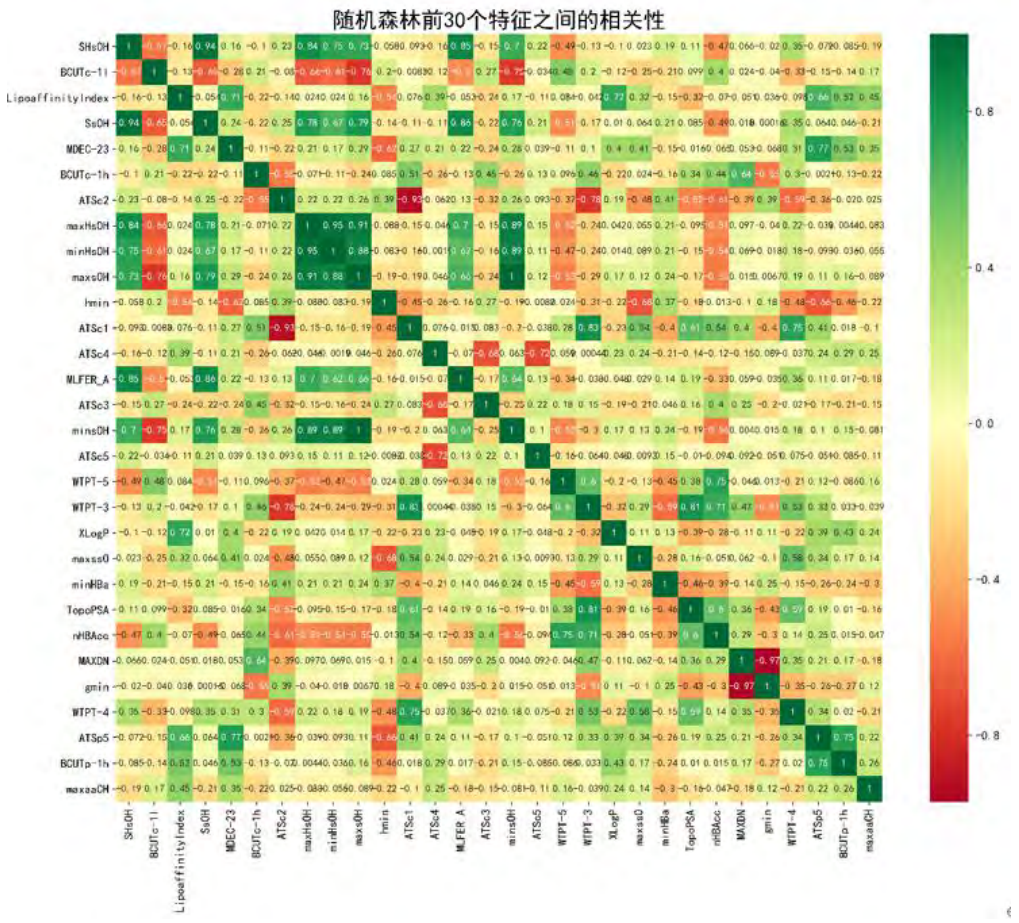


图 3-5 随机森林前 30 个特征之间的相关性

用 python 语言对随机森林的前 30 个特征选择与降维进行实现，结果如上图 3-7 所示，该图较好的呈现出了各特征之间的相关性，绿色部分表示各特征之间的正相关，红色部分表示各特征之间的负相关，颜色程度越深表示相关性越强，RF 降维法的结果显示相关性分布比较均匀。

3.3.4 方案对比

由图 3-5 和 3-7 的对比，熵值法整体偏正相关，并且该方法在筛选变量过程中和因变量 pIC50 无关。综合这两点可知熵值法挑选的 30 组特征不具有代表性，而随机森林特征选择避免了这两个缺点，因此使用随机森林挑选的 30 个特征更具有代表性。在此基础之上，本文对随机森林的 30 组变量进行相关性分析，逐一删除了图 3-7 中 30 个特征中相关性大于 0.85 的 10 个对特征中的一个，最终选择出了 20 组特征作为主要变量。

最终筛选的 20 个特征相关性分析图如下图 3-8 所示：

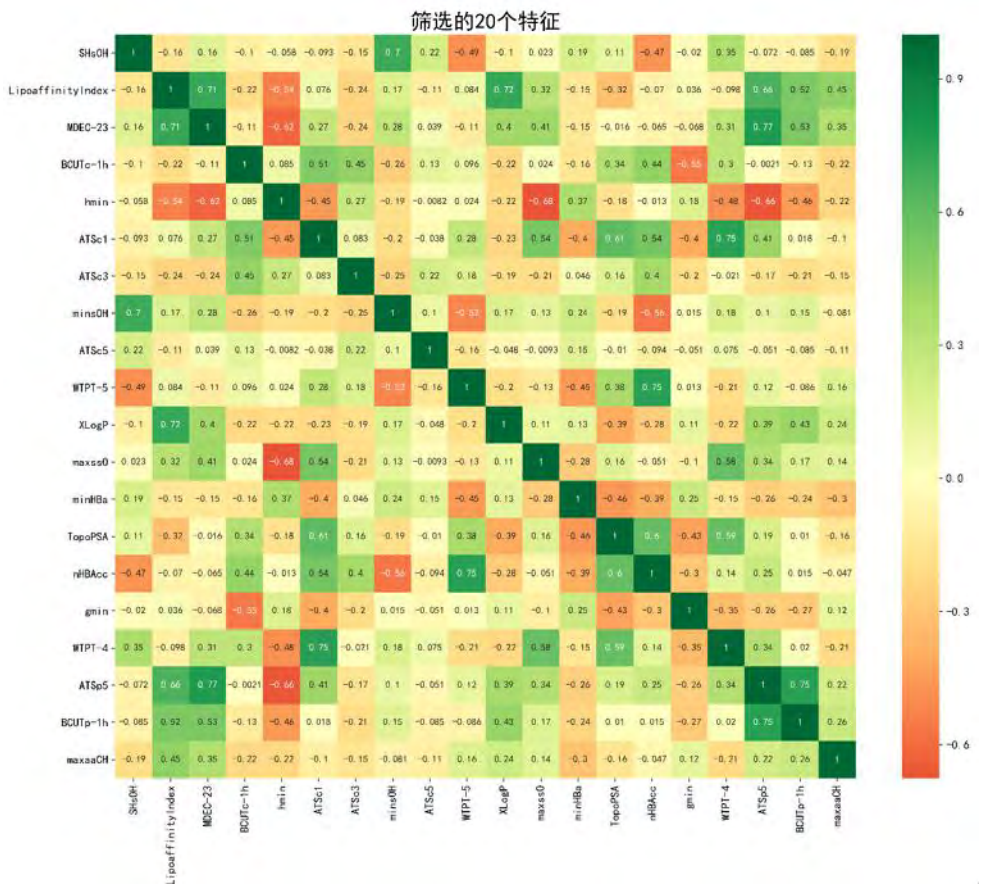


图 3-8 随机森林法筛选的 20 个特征相关性分析图

在 1974 个化合物的 729 个分子描述符信息中，前 20 个对生物活性最具有显著影响的分子描述符（即变量）及其贡献率如下表 3-1 所示：

表 3-1 对生物活性最具有显著影响的 20 个分子描述符

特征名称	贡献率	特征名称	贡献率
SHsOH	0.01045	XLogP	0.00711
LipoaffinityIndex	0.01016	maxssO	0.00708
MDEC-23	0.00953	minHBa	0.00705
BCUTc-1h	0.00833	TopoPSA	0.00702
hmin	0.00777	nHBAcc	0.00692
ATSc1	0.00771	gmin	0.00639
ATSc3	0.00739	WTPT-4	0.00632
minsOH	0.00738	ATSp5	0.00608
ATSc5	0.00733	BCUTp-1h	0.00604
WTPT-5	0.00726	maxaaCH	0.00597

在这 20 个对生物活性最具有显著影响的分子描述符（即变量）中，数量比例与累计贡献率的比例如下图 3-9 所示，由图可知，数量占比 2.7%的变量，其贡献率达到了 18%，同时说明该 20 个特征有代表性。

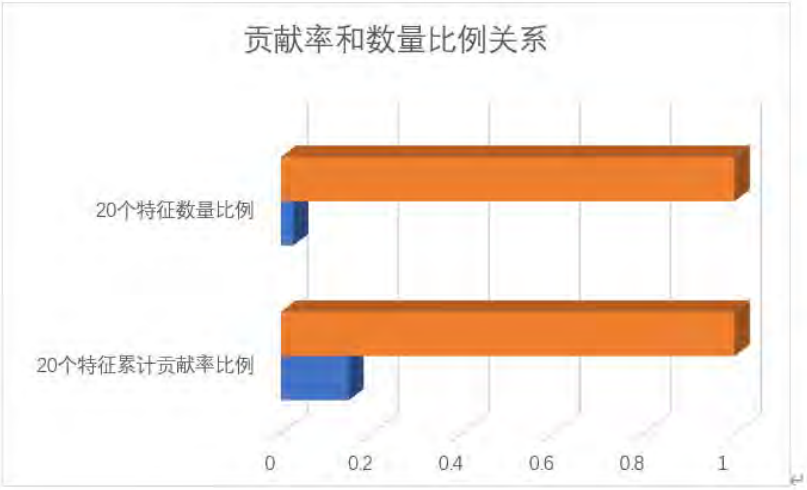


图 3-9 贡献率和数量比例关系

3.4 问题小结

问题一要求我们针对 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）。本文首先通过对附件的数据进行预处理，主要包括空缺值处理、拉依达准则判断、剔除异常数据、最值限幅法处理数据。数据预处理完之后采用随机森林和熵值法两种方法进行特征选择，分别得出各自 30 个主要变量的相关图，并对两种方法下的相关图进行对比分析，经过对比发现采用了随机森林法进行特征选择更具有代表性，然后逐一剔除随机森林前 30 个主要操作变量对中相关性大于 0.85 的变量对中的一个变量，最后获得 20 个主要操作变量。

4、问题二的模型建立与求解

4.1 问题分析

结合问题 1，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，本文采用支持向量回归、随机森林、神经网络和梯度回归四种方法分别建立化合物对 ER α 生物活性的定量预测模型，采用了随机搜索方法对基于随机森林、支持向量回归和梯度提升的回归预测模型进行超参数搜索，选出适合模型的最佳超参数并使用统一评价指标对四种模型进行综合对比分析，以确定最佳回归预测模型，使用该模型对附件一 ERα_activity.xlsx test 表中数据进行 pIC_{50} 和 IC_{50} 值的预测，并可视化预测结果。

问题二的思路流程图如下图 4-1 所示：

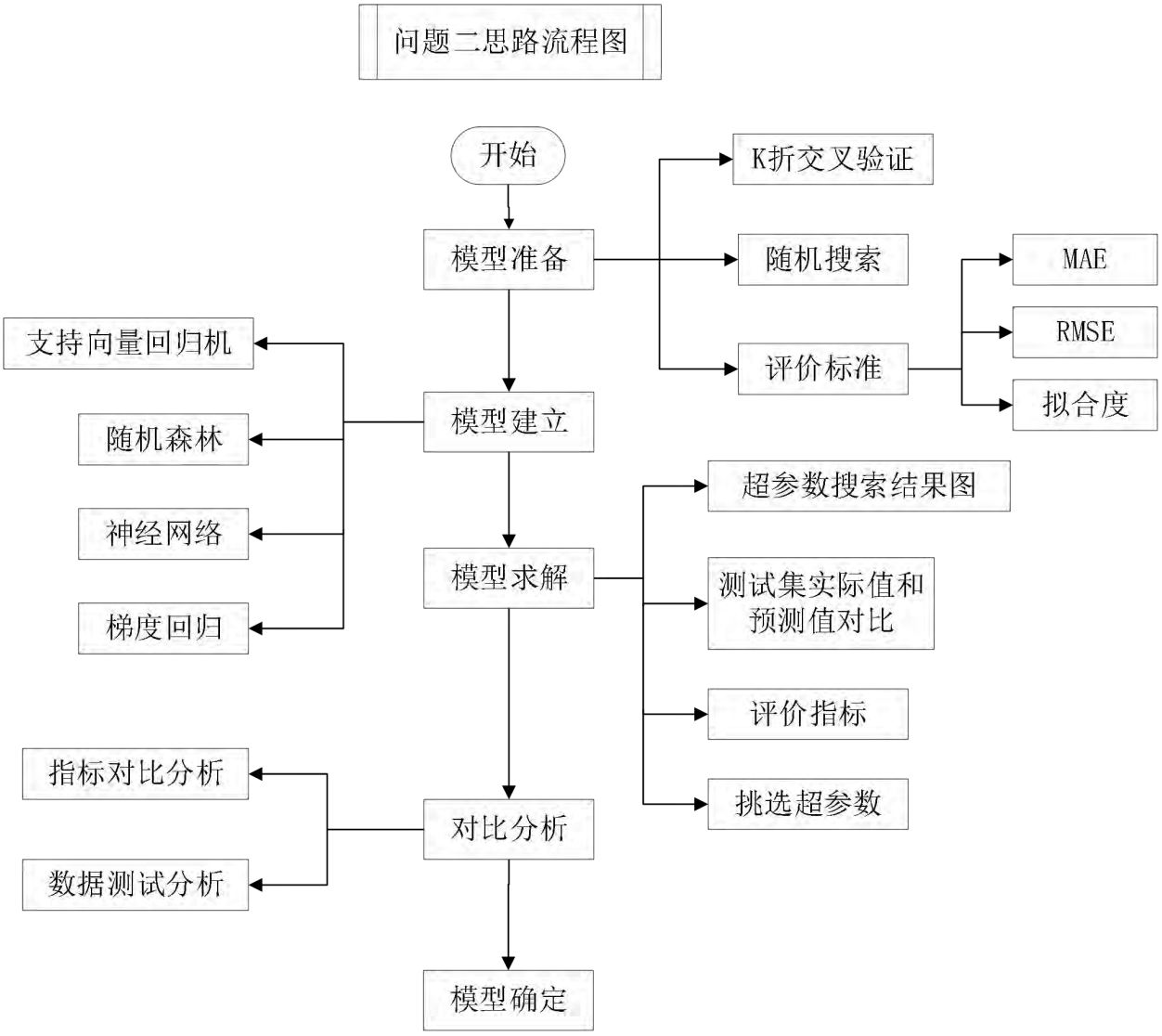


图 4-1 问题二思路流程图

4.2 模型准备

4.2.1 K 折交叉验证

K 折交叉验证使用了无重复抽样技术的好处：为了得到可靠稳定的模型，每次迭代过程中每个样本点只有一次被划入训练集或测试集的机会。在本文中运用的交叉验证法是五折交叉验证法，原理图如下图 4-2 所示，首先将数据集分为 5 等份，然后选取一份作为测试集，另外四份作为训练集，一共重复 5 次，每次选取的训练集不同。

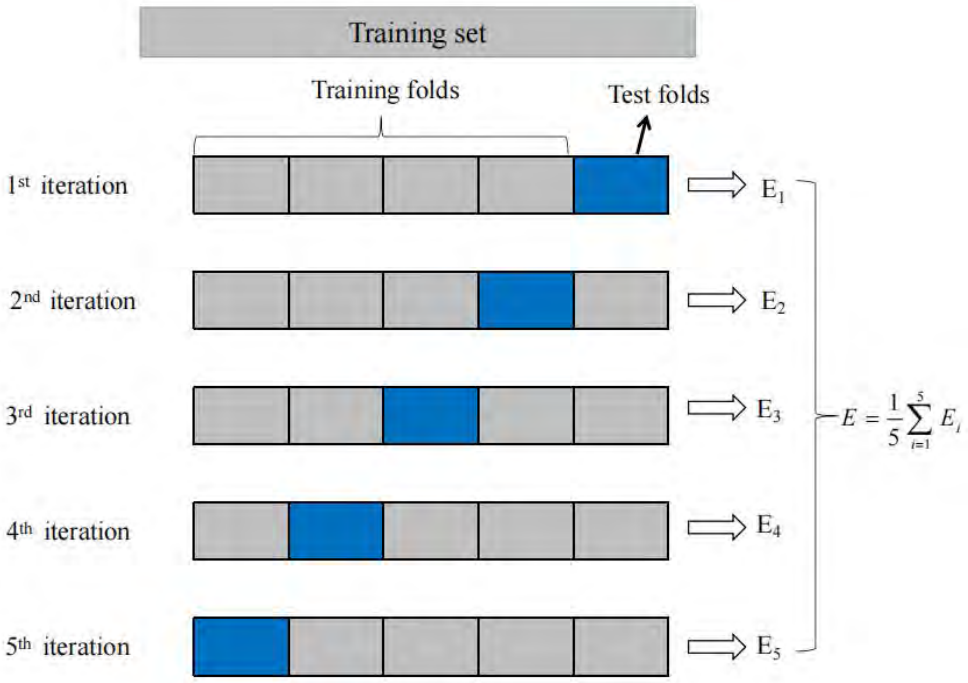


图 4-2 五折交叉验证法

4.2.2 随机搜索

随机搜索（Random Search），顾名思义就是随机的搜索，是利用随机数求极小点而求得函数近似的最优解的方法，是一组不需要优化问题梯度的数值优化方法，这种方法也被称为直接搜索、无派生或黑盒方法。基本的随机搜索算法可以描述为：

（1）在搜索空间中随机初始化 x；

（2）直到满足最终目标（比如运行的迭代次数或者要达到的适当度）。

常见的一种调参方法还有网格搜索（Grid Search），就是对网格中每个交点进行遍历，从而找到最好的一个组合，这种方法的效果不错，适用于需要对整个参数空间进行搜索的情况，但是这种方法计算代价非常大，可能会引起维度灾难。

在一些情况下，随机搜索得到的超参数组合的性能稍微差一点。随机搜索的好处在于搜索速度快，相对于很复杂的问题比较容易应用，但是这个方法的缺点在于必须通过反复多次实验来对每个问题进行特殊处理，不然容易错过一些重要的信息。

4.2.3 评价标准

（1）平均绝对误差（MAE）

平均绝对误差，又叫平均绝对离差，是所有单个观测值与算术平均值的偏差的绝对值的平均。MAE 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，具体表示如式（2）所示：

$$\frac{1}{m}\sum_{i=1}^{m}\left|(y_i-\hat{y}_i)\right| \tag{2}$$

其范围为[0,+∞)，当预测值与真实值完全吻合时等于 0，即完美模型；误差越大，该值越大。MAE 反映的就是真实误差。

（2）均方根误差（RMSE）

均方根误差，亦称为标准误差，表示如式（3）所示：

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i-\hat{y}_i)^2} \tag{3}$$

其中，y_i 为真实值，$\hat{y}_i$ 为预测值 RMSE 其实是放大了较大误差之间的差距，根号里面类似于线性回归的损失函数，我们把损失函数放在测试集上求得损失值，开根号是为了更好的描述数据。

（3）拟合度

拟合度是指回归曲线对观测值的拟合程度，度量拟合优度的统计量是可决系数 R^2。可决系数，亦称测定系数、确定系数、决定系数、可决指数。

对于 m 个样本 $(\vec{x_1},y_1),(\vec{x_2},y_2),\cdots,(\vec{x_m},y_m)$，某模型的估计值为 $(\vec{x_1},\hat{y_1}),(\vec{x_2},\hat{y_2}),\cdots,(\vec{x_m},\hat{y_m})$

计算样本的总平方和 TSS，$TSS=\sum_{i=1}^{m}(y_i-\bar{y})^2$，残差平方和 RSS，$RSS=\sum_{i=1}^{m}(\hat{y}_i-y_i)^2$。

定义 $R^2=1-RSS/TSS$，如下式（4）所示：

$$R^2=1-\frac{\sum(Y\_actual-Y\_predict)^2}{\sum(Y\_actual-Y\_mean)^2} \tag{4}$$

其中，分子使我们训练出来的模型预测的所有误差，分母表示期望样本误差。R^2 越大，拟合效果越好。R^2 的最优值为 1，若预测模型为随机值，R^2 有可能为负值；若预测值恒为样本期望，R^2 为 0。

4.3 模型建立

4.3.1 支持向量回归

支持向量机（SVM）是一种经典的机器学习算法，在小样本数据集的场景中有较为广泛的应用。SVM 主要是将数据进行分类，其模型思想是：假设平面中有两种类型的点，一种是红色一种是蓝色，要求我们将这两种类型的点分开，取一条最优的分界线，既能把两种类型分开，还使得两类样本点离分界线最近的点离分界线的距离尽可能远。对于在高维空间中，不像二维空间中的一条直线，把高维空间中的分界线叫做超平面，而真正决定分割超平面作用的点叫做支持向量。

支持向量回归（SVR）是 SVM 对回归问题的一种运用，主要是升维之后，在高维空间中构造线性决策函数来实现线性回归。SVR 模型的模型函数是非线性高斯核，可以表示为下式（5）：

$$\phi_{\gamma}(x,\ell)=\exp(-\gamma\left\|x-\ell\right\|)^{2} \tag{5}$$

这是一个从 0（离地标差的非常远）到 1（跟地标一样）变化的钟形函数。

4.3.2 神经网络

深度神经网络（DNN）是一种模仿大脑神经系统对外来信息处理的数学算法模型，网络中的每一个节点可以视为一个“神经元”，通过“神经元”之间的相互连接，共同组成一个高效而完整的信息处理网络。只有相邻层的“神经元”之间才会互相连接，每一层的“神经元”的数量不尽相同，神经网络通过对输入信息的层层传递与处理，得到自己想要的结果。深度神经网络包括输入层、隐藏层和输出层，网络的深度取决于隐藏层的数目，网络的宽度取决于隐藏层“神经元”的数目，一般越深的网络越难训练，同时其收敛速度也会越慢，训练效果也会不理想，在网络深度加深的同时增加网络的宽度，会有助于网络的收敛。

神经网络算法示意图如下图 4-3 所示：

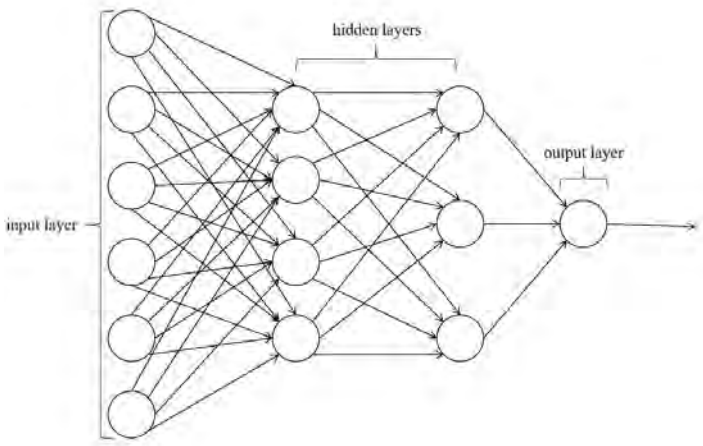


图 4-3 神经网络算法示意图

本文中构建基于深度神经网络的生物活性定量预测模型的具体步骤为：

1、利用问题 1 中挑选出的 20 个对生物活性最具有显著影响的分子描述符数据和附件 1 提供的 pIC50 数据（这里使用的数据均是训练数据）按照 8:2 的比例划分为训练集和测试集；

2、构建深度神经网络模型，该模型主要由输入层、隐藏层和输出层组成，输入层和隐藏层都由 Linear-ReLU-Dropout 结构组成，其中 Dropout 的比例设置为 0.2（此次训练经验值），其中隐藏层的层数设置为 6，输出层是由一个全连接层构成，直接输出深度神经网络对自变量（20 个分子描述符数据）的预测值；

3、初始化损失函数和优化器，训练用损失函数使用 MSE 损失函数，优化器可选择 SGD 和 Adam 优化器等；

4、定义训练次数并开始模型的训练。

4.3.3 梯度提升

梯度提升（GBRT），是一种迭代的回归树算法，由多棵回归树组成，所有树的结论累加起来得到最终结果。其核心就在于，每一棵树是从之前所有树的残差中来学习的，这个残差就是一个加预测值后能得真实值的累加量。GBRT 主要有两个部分组成：回归树（RT）和梯度提升（GB）。（1）RT：GBRT 是迭代的决策树算法，决策树分为两类，回归树和分类树，分类树用于分类标签值，回归树用来预测实际值。GBRT 也是一种迭代的回归树算法，由多棵回归树组成，所有树的结论累加起来得到最终结果。（2）GB：Boosting 通过迭代多棵树来共同决策，核心是每一棵树都是学习之前所有树的结论和残差。

梯度提升回归树模型总体来说时非常不错的，它的优势是比较精确，因为在提升过程中是在不断修正前面的决策树。

4.3.4 随机森林

随机森林（Random Forest，简称 RF），是一种新兴起的、高度灵活的、基于树的机器学习算法，该算法利用多棵树的力量来进行决策。森林中的每棵树并不一样，每棵树都是被随机创造的，每棵树中的每一个节点都是待选特征的一个随机子集，所有树的输出结果整合起来就是森林最后的输出结果。

随机森林重点是“随机”两个字，它包括两个方面：

（1）训练数据的随机选取。即用于训练单棵树的数据应该是随机、有放回的从全部数据中选取和原始数据集相同的数据量

（2）待选特征的随机选取。使得森林中的各棵树都能够彼此不同，提示系统的多样性，从而提升分类性能。

构造随机森林的步骤可描述为：

Step1：假如有 N 个样本，则有放回的随机选择 N 个样本。用选择好的 N 个样本来训练一个决策树，作为决策树根节点处的样本。

Step2：当每个样本有 M 个属性时，在决策树的每个节点需要分裂时，随即从 M 个属性中选取 m 个属性，满足 m<<M。然后从 m 个属性中采用某种策略（比如信息增益）来选择一个属性作为该节点的分裂属性。

Step3：决策树形成过程中每个节点都要按照 step2 来分裂，一直到不能再分裂为止。注意整个决策树形成过程中没有进行剪枝。

Step4：按照 step1-3 建立大量的决策树，这样就构成了随机森林。

随机森林算法示意图如下图 4-4 所示：

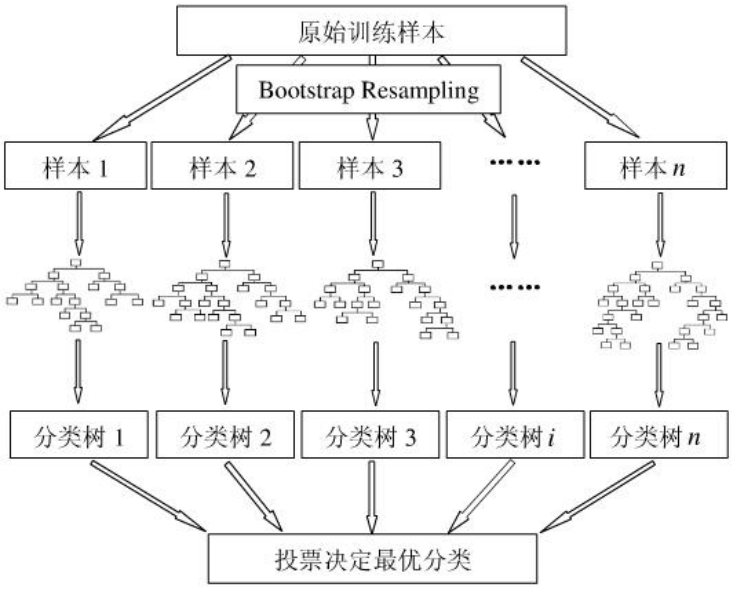


图 4-4 随机森林算法示意图

4.4 模型求解

本文首先对 SVR、DNN、GBRT 和 RF 模型进行超参数的确定，使用随机搜索的方式对 SVR、GBRT 和 RF 模型进行 100 次一定范围内的最佳超参数搜索，按照经验调参法确定 DNN 模型的超参数。使用确定的超参数建立 4 个回归预测模型，采用 5 折交叉验证验证模型的鲁棒性并分别对测试集中数据进行预测，使用 RMSE、MAE 和拟合度作为模型的评价指标，分别可视化四个模型的预测情况，进一步对比分析模型的优劣，以确定适合本问题的最佳回归预测模型。

4.4.1 支持向量回归模型求解

对 SVR 模型最佳超参数进行随机搜索，先在较大范围（C:1-100,gamma：0.0001-0.1）内进行粗略搜索，根据搜索图缩小搜索范围（C:1-10,gamma：0.001-0.01）内进行精确搜索，搜索图如下图 4-5 所示：

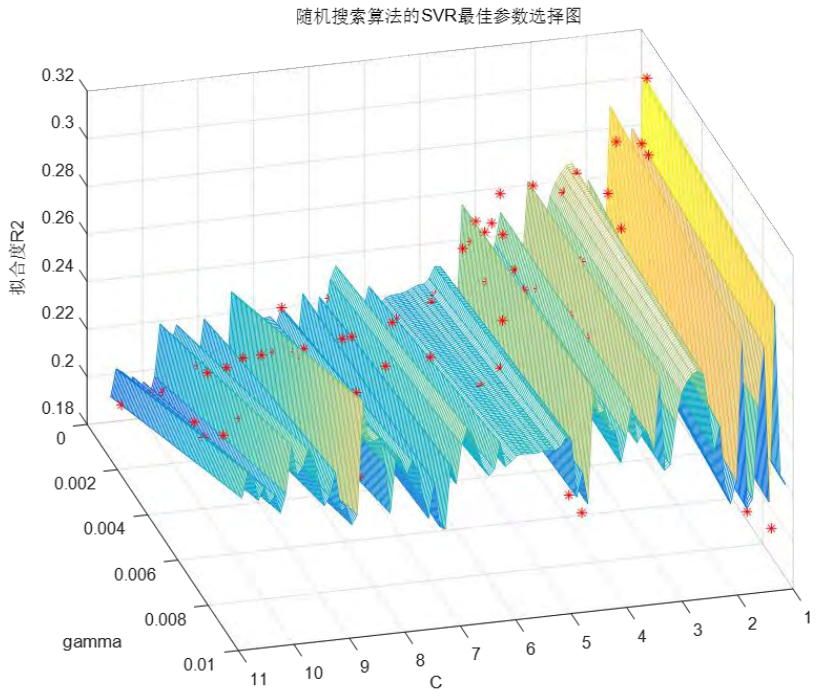


图 4-5 随机搜索算法的 SVR 最佳参数选择图

红色点为随机搜索对应的位置，平面坐标是两个超参数，纵坐标是是模型拟合度，由图可知，C 和 gamma 值较小时有较高的拟合度，但是最佳拟合值是：0.367，可见模型改进空间较大。随机搜索的结果如下表 4-1 所示：

表 4-1 随机搜索结果

C	Gamma
1.2542	0.001282

画出测试集前 30 组数据 pIC50 的实际值和预测值的对比图如下图 4-6 所示：

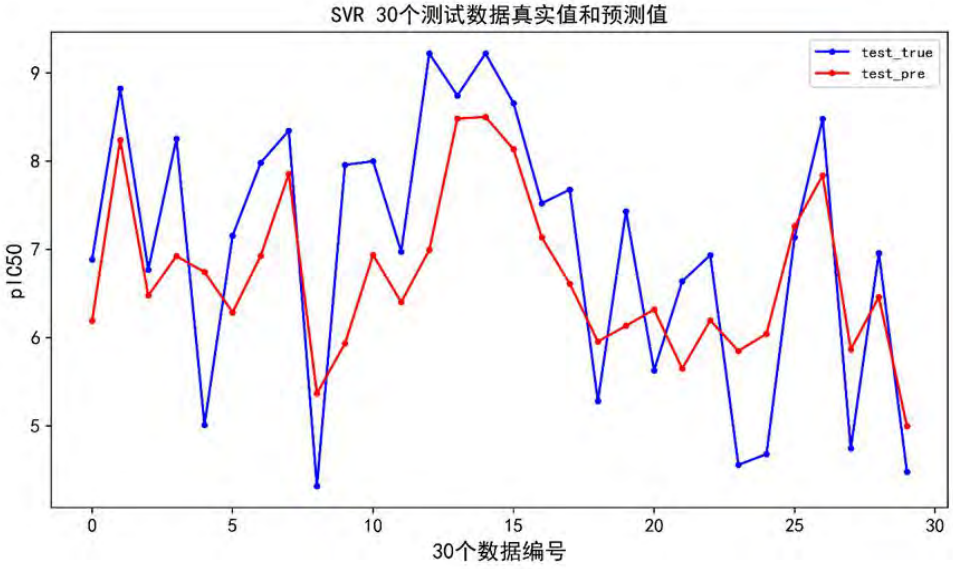


图 4-6 SVR30 个测试数据真实值和预测值结果图

如上图所示，可以直观的看到，支持向量回归模型预测的结果和实际值整体走势相似，但是数值之间的差距较大。加上拟合度较低，因此模型改进空间较大。

4.4.2 深度神经网络模型求解

深度神经网络的训练时间成本较大，没有进行超参数的随机搜索，而是根据经验来调节超参数。使用 pytorch 对深度神经网络进行训练，训练过程中迭代 1000 次，训练结果如下图所示。

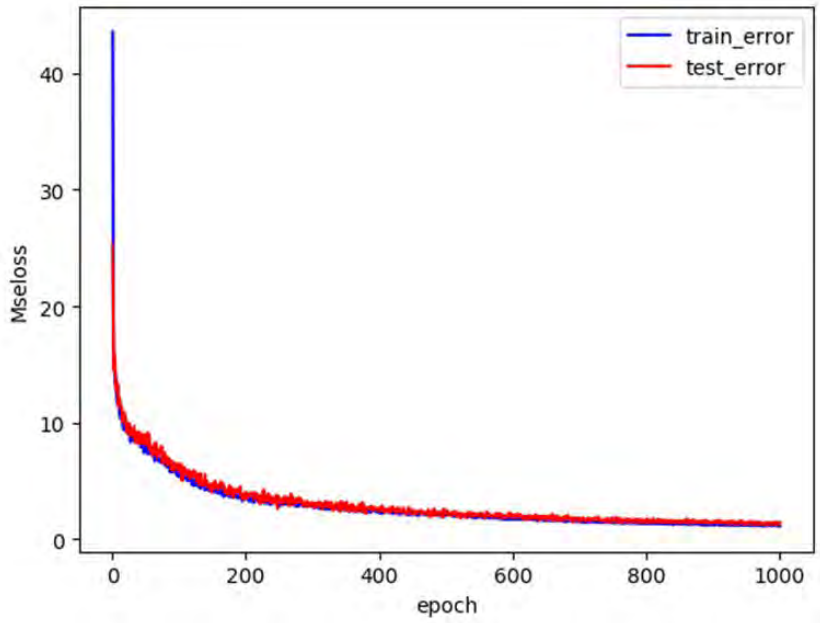


图 4-7 基于 MSE 的训练损失图

由上图 4-7 可以看出训练集和测试集的 MSE 损失基本相同，没有过拟合或欠拟合，最佳拟合度为 0.337，改进空间较大，而且运行时间较久。

画出测试集前 30 组数据 pIC50 的实际值和预测值的对比图如下图 4-8 所示：

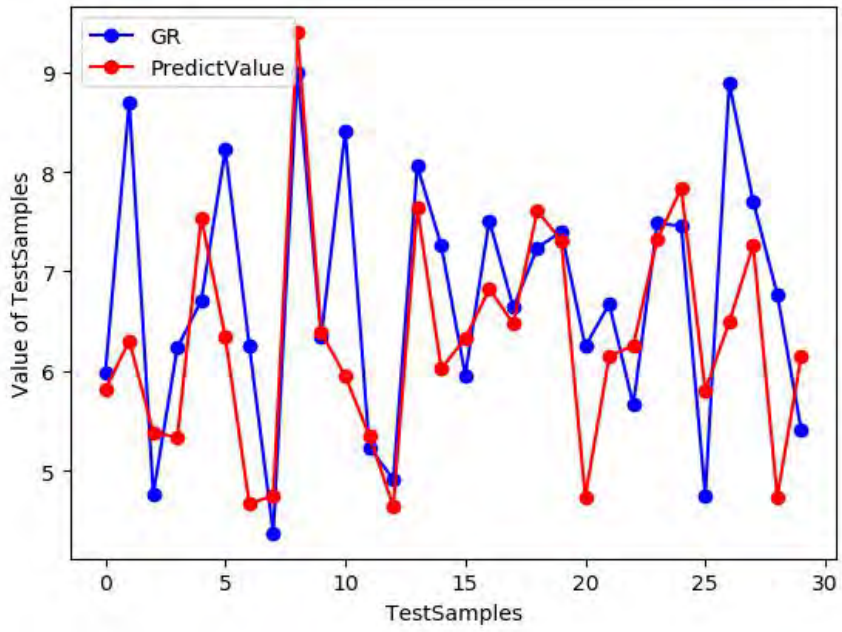


图 4-8 DNN 算法下 pIC50 的实际值和预测值的对比图

根据测试集 30 组数据显示，DNN 模型对很多突变数据拟合效果较差，考虑到模型的整体拟合度较低，参数调整困难，训练时间成本高，尽管 DNN 近年来在很多领域的表现很出色，但是本文训练得到的该 DNN 模型依然有较大改进空间。

4.4.3 梯度提升回归模型求解

对 GBRT 模型最佳超参数进行随机搜索，先在较大范围（max_depth:1-100, n_estimators：1-500）内进行粗略搜索，根据搜索图缩小搜索范围（max_depth:2-20, n_estimators：20 -200）内进行精确搜索，搜索图如下图 4-9 所示：

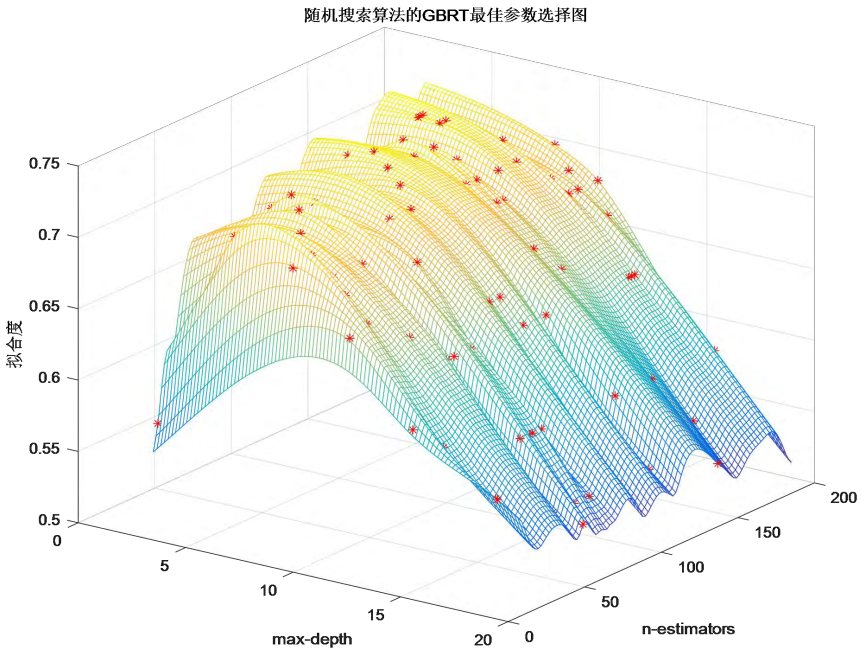


图 4-9 随机搜索算法的 GBRT 最佳参数选择图

红色点为随机搜索对应的位置，平面坐标是两个超参数，纵坐标是是模型拟合度，由图可知， n_estimators 值在一定的范围内越大，模型拟合度越高，max_depth 在 5 附近时达到最大，整体最佳拟合值是：0.72，模型拟合较前两种方法有较大提升。随机搜索的结果如下表 4-2 所示：

表 4-2 随机搜索结果

max_depth	n_estimators=123
5	123

画出测试集前 30 组数据 pIC50 的实际值和预测值的对比图如下图 4-10 所示：

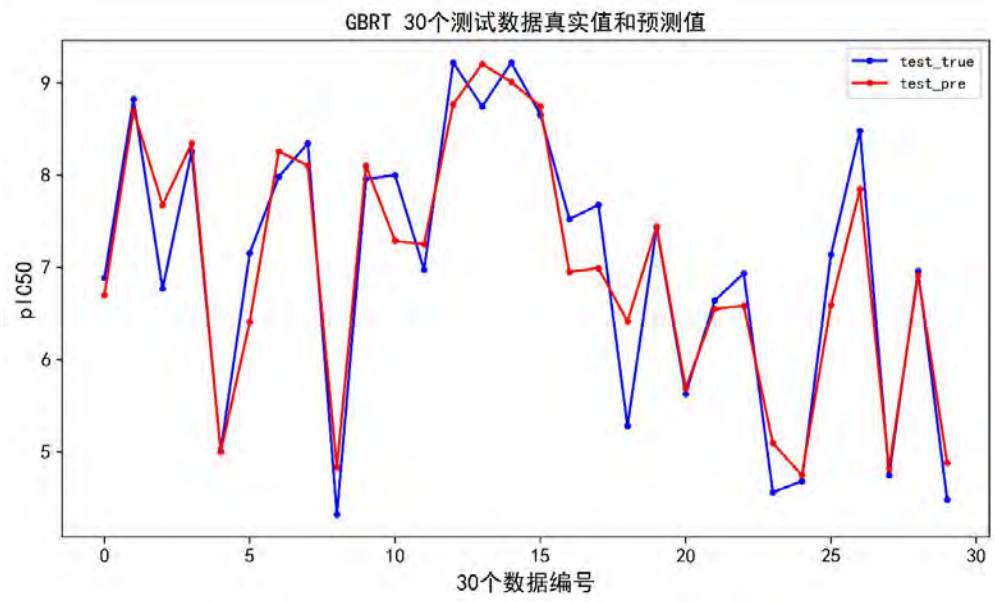


图 4-10 GBRT30 个测试数据真实值和预测值对比图

如上图所示，可以直观的看到，梯度提升模型预测的结果和实际值拟合效果很好，加上整体拟合度达到了 0.72，提升较大，可以考虑作为候选模型，但是该模型训练时间久：100 次随机搜索用时 323s，提升空间较大，故进一步寻找较优模型。

4.4.4 随机森林模型求解

对 RF 模型最佳超参数进行随机搜索，先在较大范围（max_features:1-50, n_estimators：1 -1000）内进行粗略搜索，根据搜索图缩小搜索范围（max_features:1-8, n_estimators：1-200）内进行精确搜索，搜索图如下图 4-11 所示：

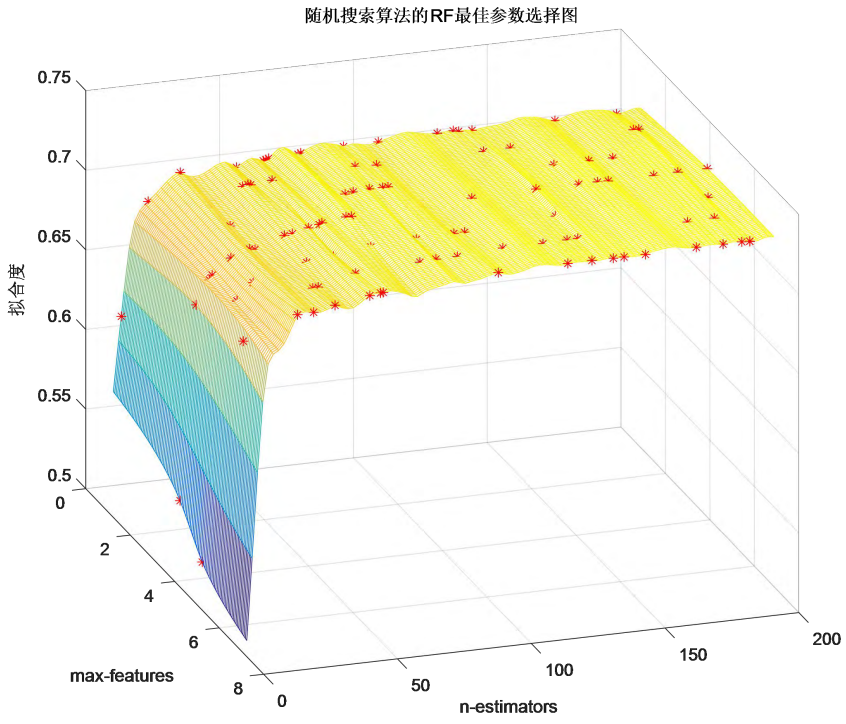


图 4-11 随机搜索算法的 RF 最佳参数选择图

红色点为随机搜索对应的位置，平面坐标是两个超参数，纵坐标是是模型拟合度，由图可知，n_estimators 在一定范围内，越大时对应拟合度越高，模型整体最佳拟合值是：0.74，可见模型改进空间较大。随机搜索的结果如下表 4-3 所示：

表 4-3 随机搜索结果

max_features	n_estimators
6	185

画出测试集前 30 组数据 pIC50 的实际值和预测值的对比图如下图 4-12 所示：

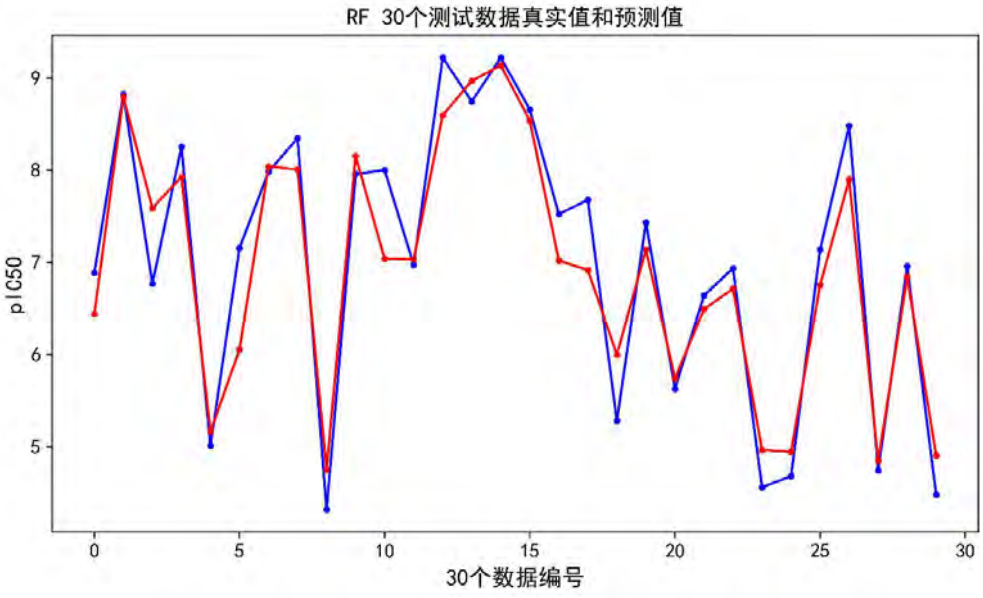


图 4-12 RF30 个测试数据真实值和预测值对比图

如上图 4-12 所示，可以直观的看到，随机森林模型预测的结果和实际值拟合效果很好，加上整体拟合度达到了 0.74，提升很大，可以考虑作为候选模型，而且该训练时间相比较前几个模型很短：100 次随机搜索用时 136s，因此该模型可以作为预测模型。

4.5 对比分析

根据上述评价标准中的评价指标对四种模型的 RMSE、MAE 以及拟合度进行计算，得到结果如下表 4-4 所示，再根据四种模型的评价指标绘制出统计图进行清晰可观的对比分析，绘制的模型对比图如下图 4-13、4-14 所示：

表 4-4 四种算法的不同评价指标

算法名称	测试 RMSE	测试 MAE	拟合度 R^2	时间/s
SVR	1.1601	0.9269	0.3672	809.75
ANN	1.1898	0.9229	0.3370	871.29
GBRT	0.7683	0.5691	0.7194	323.33
RF	**0.7412**	**0.5511**	**0.7367**	**136.30**

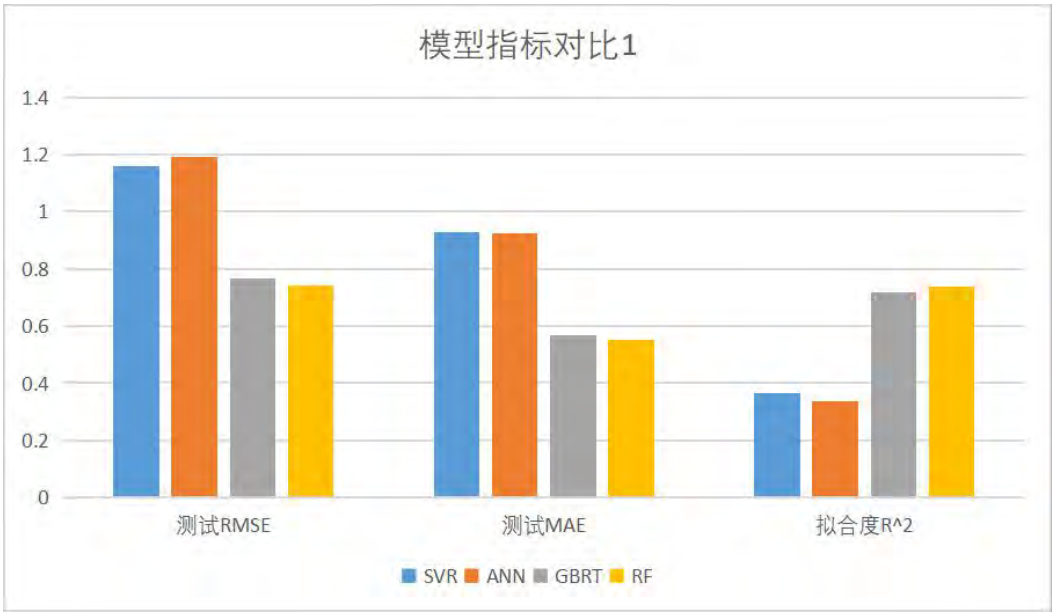


图 4-13 四种模型的评价指标对比统计图

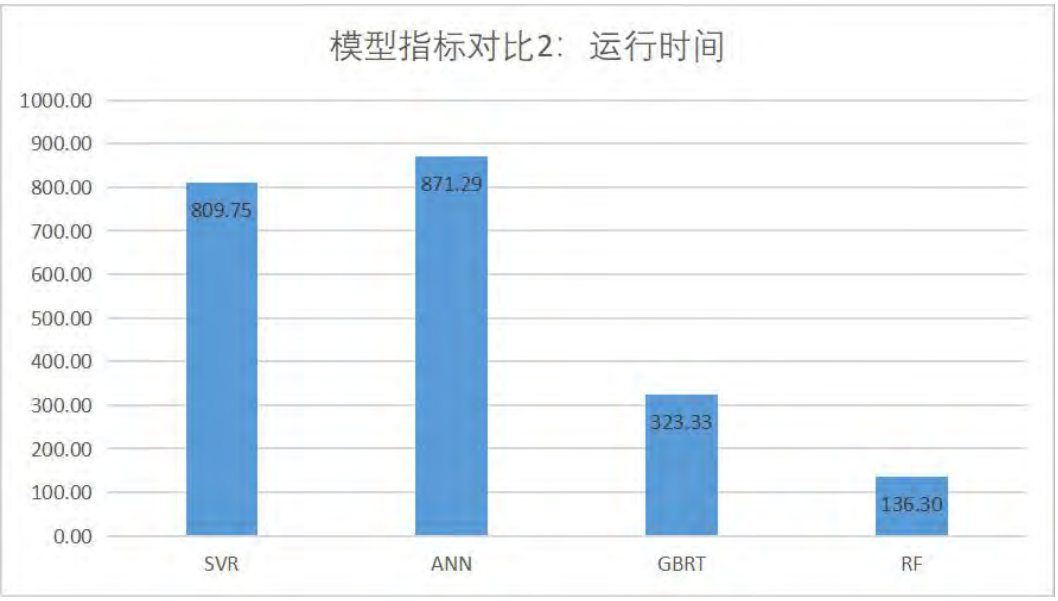


图 4-14 四种模型的训练时间对比统计图

四种评价指标中 RMSE 和 MAE 值越小模型预测值越接近实际值；拟合度的 R^2 越大，模型对数据集拟合度越好。

综合图 4-13 和图 4-14 中对四种模型的对比分析，可以得出：基于随机森林和梯度回归的预测模型在测试集上的 RMSE、MAE 值比较小，基于随机森林的预测模型的训练时间远小于基于梯度回归的预测模型，所以本文决定使用基于随机森林的预测模型作为化合物对 *ERα* 生物活性的定量预测模型。

使用基于随机森林的回归预测模型，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，结果如下表 4-5 所示，同时用 python 实现了 test 表中 50 组测试数据的 pIC50 预测值的可视化，如下图 4-15 所示。

表 4-5 测试集中 50 个化合物的预测值

IC50_nM	pIC50	IC50_nM	pIC50	IC50_nM	pIC50
22.3637	7.6505	199.0116	6.7011	84.9715	7.0707
93.8797	7.0274	3.8079	8.4193	87.8722	7.0561
62.5700	7.2036	296.8831	6.5274	77.5386	7.1105
65.0982	7.1864	75.2614	7.1234	1339.1910	5.8732
27.2158	7.5652	76.5164	7.1162	1298.0909	5.8867
62.6458	7.2031	58.7667	7.2309	1188.8509	5.9249
44.2081	7.3545	370.5344	6.4312	1157.7162	5.9364
51.0341	7.2921	217.0769	6.6634	1038.2414	5.9837
155.0533	6.8095	26.4067	7.5783	1216.3853	5.9149
76.5921	7.1158	89.4121	7.0486	1188.8509	5.9249
79.0102	7.1023	74.3407	7.1288	89.6552	7.0474
64.9390	7.1875	1807.7584	5.7429	110.6684	6.9560
61.1648	7.2135	3951.0049	5.4033	145.3806	6.8375
41.7297	7.3796	4164.3796	5.38045	215.5260	6.6665
32.6998	7.4855	4260.8860	5.3705	6.0250	8.2200
50.1178	7.3000	4358.6571	5.36065	4488.0580	5.3479
72.6095	7.1390				

画出测试集中 50 组化合物预测值的走势图，如图 4-15 所示：

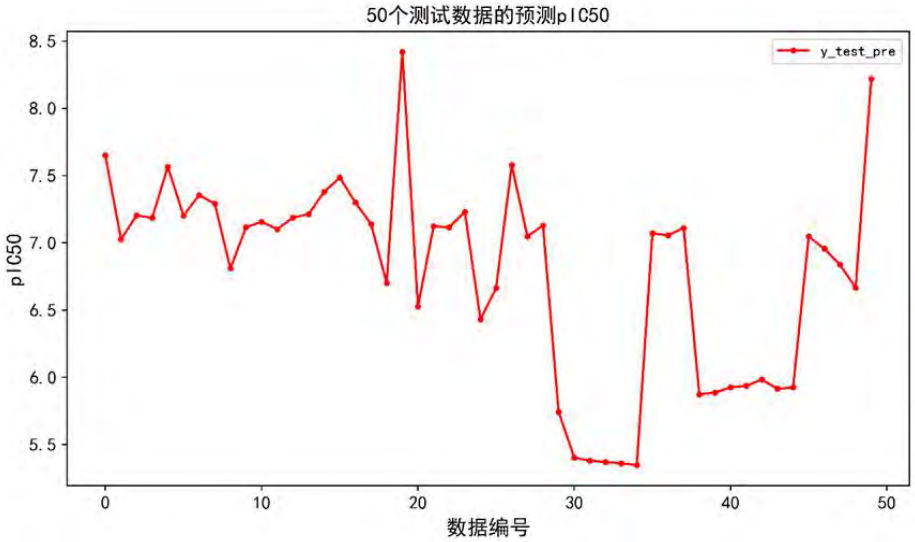


图 4-15 50 个测试数据的 pIC50 预测值折线图

4.7 问题小结

通过对比四种模型测试时的具体表现，可以得出：基于随机森林的回归预测模型的效果最好，所以采用该模型最为解答第二问的模型，并使用该模型对文件“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。

5、问题三的模型建立与求解

5.1 问题分析

问题三要求针对文件“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，使用第一问中数据预处理之后剩余的 359 个特征数据作为自变量，五个化合物分类数据作为因变量。然后使用 ROC、AUC 和准确率作为评价指标；其次在问题二模型建立的基础上新增 SGD 模型去掉 GBRT 模型，接着分别用 SVM、SGD、DNN、RF 四种模型算法构建化合物的 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型并比较它们的优劣性，确定最优模型。最后用所构建的 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。

具体思路流程图如图 5-1 所示：

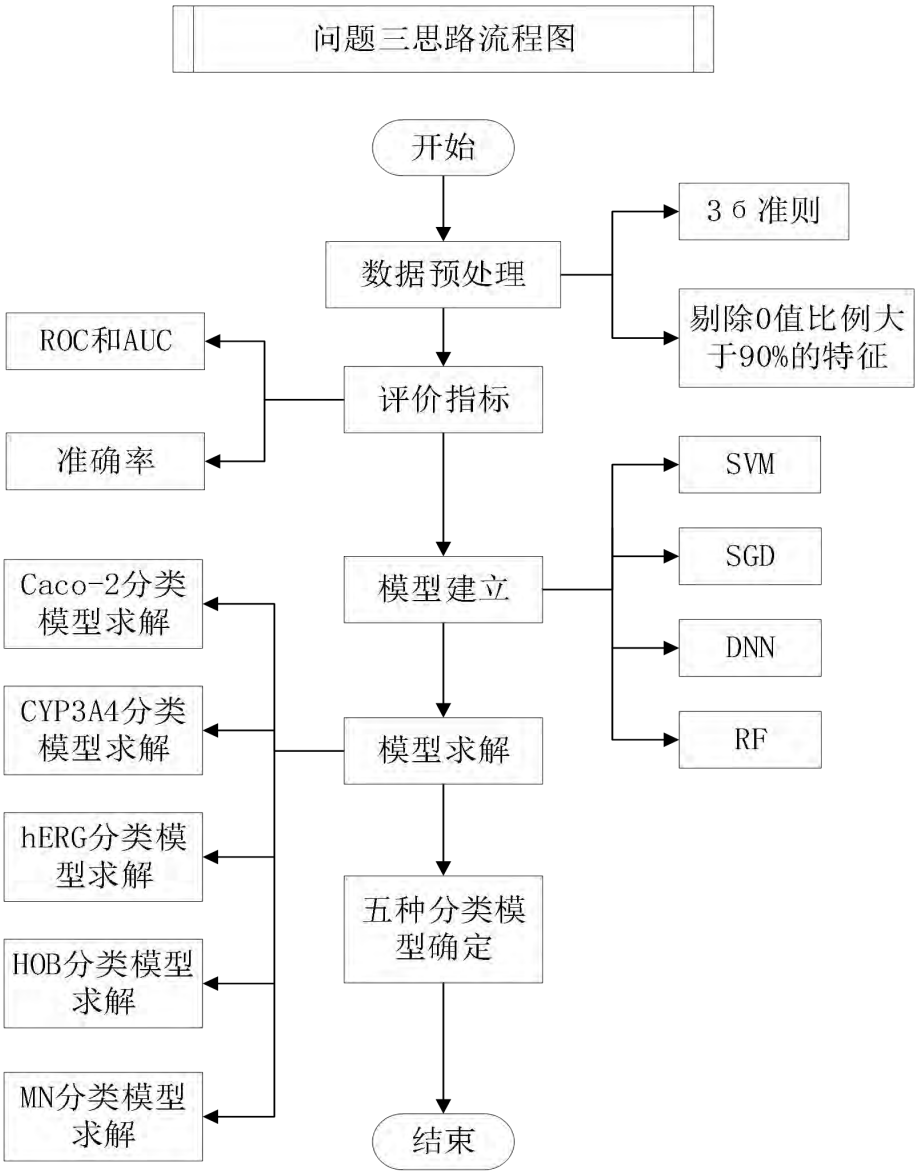


图 5-1 问题三思路流程图

5.2 数据预处理

本文对附件“Molecular_Descriptor.xlsx”中提供的 1974 个化合物的 729 个分子描述符数据进行预处理，首先采用依拉达（3σ）准则，剔除特征值不在 $\mu \pm 3\sigma$ 范围内的异常值，提高数据的稳定性，然后对数据进行排查筛选，剔除特征中 0 值比例大于 90%的数据，该操作剔除了 344 个特征，数据预处理完毕。

5.3 评价指标

5.3.1 ROC 和 AUC

(1)ROC

ROC，亦称为受试者工作特征曲线。在二分类问题中，ROC 曲线的每一个点都代表一个阈值，分类器给每个样本一个 score，该分数若大于阈值被定义为正样本，小于阈值则被定义为负样本。

ROC 有个很好的优势：当测试集中的正负样本分布变化时，ROC 曲线能够保持不变。在实际的数据集中经常会出现分类不平衡现象，也就是负样本比正样本多很多或者少很多，而且测试数据中的正负样本分布也可能随着时间变化。

ROC 曲线的特点：阈值取值越多，ROC 曲线越平滑；ROC 曲线上的点越靠近左上角越好。

(2)AUC

AUC，指的是 ROC 曲线下的面积，该指标能较好的概括不平衡样本分类器的性能，它的直观含义是任意取一个正样本和负样本，正样本得分大于负样本的概率。AUC 值为 ROC 曲线所覆盖的区域面积，显然，AUC 值越大，反映出正样本的预测结果更加靠前，分类器分类效果越好。

另外，P/R 曲线和 ROC 曲线都是用于分类的评价指标，一般情况下，P/R 曲线用于检索，而 ROC 曲线一般用于分类、识别等。由于在实际问题中，正负样本数通常很不均衡，若选择不同的测试集，P/R 曲线的变化就会非常大，而 ROC 曲线则能够更加稳定地反映模型本身的好坏。

在该附件提供的 1974 个化合物的 ADMET 数据中，Caco-2、CYP3A4、hERG、HOB、MN 五种化合物出现的频数如下图 5-2 所示，由图表可以看出该数据集中的正负样本不均衡，所以在 P/R 曲线和 ROC 曲线的选择中本文选择 ROC 曲线来作为评价指标评价指标之一，放弃了使用 P/R 曲线。

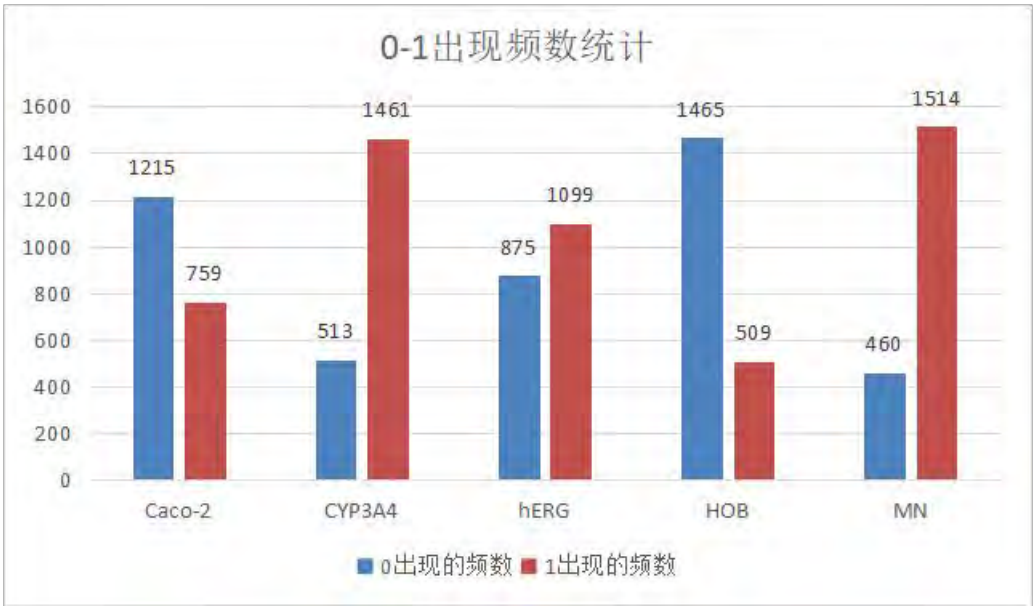


图 5-2 数据集中 0/1 出现频数统计图

5.3.2 准确率

准确率，表示被预测正确的比例，准确率是我们最常见的评价指标。对于准确率和召回率的选择，二者是矛盾的两个指标，召回率主要强调把样本中的正例样本挑出来，而准确率是强调被挑出来的样本要尽可能的准确，考虑到本题药物的使用具有很低的容错性，所以挑选的样本准确率要尽可能大，因此准确率做为本问的另一个评价指标。

可表示如下式（6）所示：

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

其中，TP 表示正确地预测为正例，TN 表示正确地预测为反例，FP 表示错误地预测为正例，FN 表示错误地预测为反例。准确率就是分类预测正确的样本数除以所有的样本数，通常来说，准确率越高，分类器越好。准确率是一个很好很直观的评价指标，但是有时候准确率高并不代表这个算法就一定好，比如在正负样本不平衡的情况下，准确率这个指标有一定的缺陷。

5.4 模型建立

5.4.1 支持向量机

支持向量机，简称 SVM，SVM 是一种经典的机器学习算法，在小样本数据集的场景中有较为广泛的应用。SVM 主要是将数据进行分类，其模型思想是：假设平面中有两种类型的点，一种是红色一种是蓝色，要求我们将这两种类型的点分开，首先想到的是从两类样本中间画一条直线，但是这种情况下两类样本点离分界线都很近，如果继续输入一个新的样本点，可能会造成分错类的现象，因此要取一条最优的分界线，既能把两种类型分开，还使得两类样本点离分界线最近的点离分界线的距离尽可能远。

5.4.2 随机梯度下降法

梯度法思想的三要素：出发点、下降方向、下降步长。

随机梯度下降法（Stochastic Gradient Descent），简称 SGD。对于训练速度来说，SGD 一次迭代只用一条随机选取的数据，虽然迭代次数会很多，但是一次学习时间非常快；对于准确度来说，SGD 仅仅用一个样本决定梯度方向，导致求解可能不是最优；对于收敛速度来说，SGD 一次迭代一个样本，导致迭代变化很大，不能很快的收敛到局部最优解。不过，如果目标函数有盆地区域，SGD 会使优化的方向从当前的局部极小值点跳到另一个更好的局部极小值点，对于非凸函数最终收敛于一个较好的局部极值点，甚至全局极值点。

随机梯度下降法的公式归结为通过迭代计算特征值从而求出最合适的值，θ的求解公式如式（7）：

$$\theta = \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \tag{7}$$

其中，α是下降系数，即步长，学习率，通俗的说就是计算每次下降的幅度大小，系数越大每次计算的差值越大，系数越小则差值越小，但是迭代计算的时间也会延长，θ的初值可以随机赋值。

5.4.3 神经网络

深度神经网络（DNN）是一种模仿大脑神经系统对外来信息处理的数学算法模型，网络中的每一个节点可以视为一个“神经元”，通过“神经元”之间的相互连接，共同组成一个高效而完整的信息处理网络。

5.4.4 随机森林

随机森林（Random Forest，简称 RF），是一种新兴起的、高度灵活的、基于树的机器学习算法，该算法利用多棵树的力量来进行决策。森林中的每棵树并不一样，每棵树都是被随机创造的，每棵树中的每一个节点都是待选特征的一个随机子集，所有树的输出结果整合起来就是森林最后的输出结果。

5.5 模型求解

对于模型的求解，分别对五类化合物进行四种模型的求解、评价和预测。求解使用挑选的 359 个变量作为自变量，五个化合物各自的分类数据作为因变量。模型的评价部分，分别用 ROC、AUC、准确率和模型训练时间四个指标作为评价指标确定五个化合物各自的最佳模型。最后，对给出的 50 组测试集数据进行预测。

5.5.1 Caco-2 分类模型求解

使用 python 对四种模型（SVM、SGD、DNN、RF）实现在化合物 Caco-2 变量下的 ROC 曲线绘制，对比图如下图 5-3 所示：

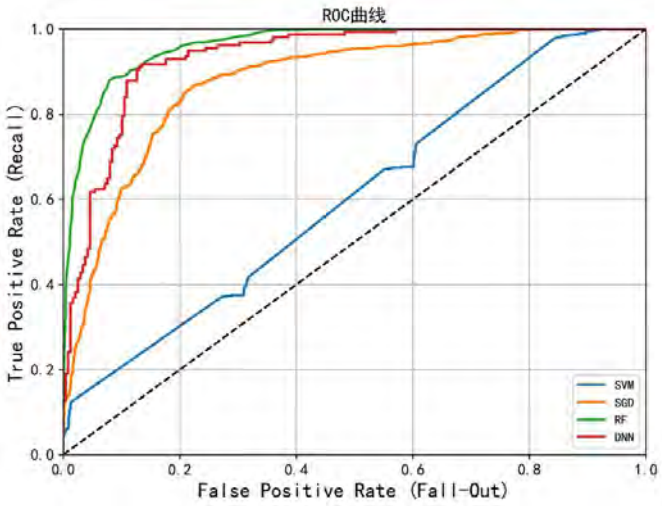


图 5-3 化合物 Caco-2 在四种预测模型下的 ROC 曲线对比图

由图 5-3 可以观察出，随机森林 RF 的 ROC 曲线最靠左，AUC 最大，预测效果最好。而支持向量机 SVM 的 ROC 曲线最靠右，AUC 最小，预测效果最差。根据上述评价标准中的评价指标对化合物 Caco-2 在四种模型下的 AUC 值、训练集准确率以及测试集准确率进行计算，得到结果如下表 5-1 所示，再根据四种模型的评价指标绘制出统计图进行清晰可观的对比分析，绘制的模型对比图如下图 5-4 所示：

表 5-1 化合物 Caco-2 在四种模型下的评价指标结果

Caco-2	AUC 值	训练集准确率	测试集准确率
SVM	0.6051	0.6466	0.6077
SGD	0.8780	0.8095	0.7618
DNN	0.9335	0.8341	0.8486
RF	**0.9626**	**0.8980**	**0.8529**

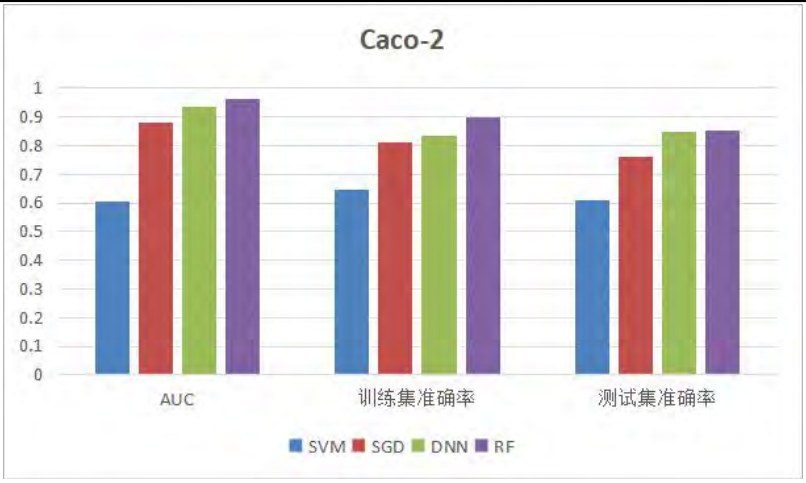


图 5-4 化合物 Caco-2 在四种模型下的评价指标对比图

根据上述评价指标我们已经知道：ROC 曲线上的点越靠近左上角越好；AUC 值越大，反映出正样本的预测结果更加靠前，分类器分类效果越好；准确率一般情况下都是越高越好。因此在图 5-4 中我们可以看出 RF 的 AUC 值最高，训练集准确率与测试集准确率最高，用随机森林算法构建化合物 Caco-2 的分类预测模型最优。

5.5.2 CYP3A4 分类模型求解

使用 python 对四种模型（SVM、SGD、DNN、RF）实现在化合物 CYP3A4 变量下的 ROC 曲线绘制，对比图如下图 5-5 所示：

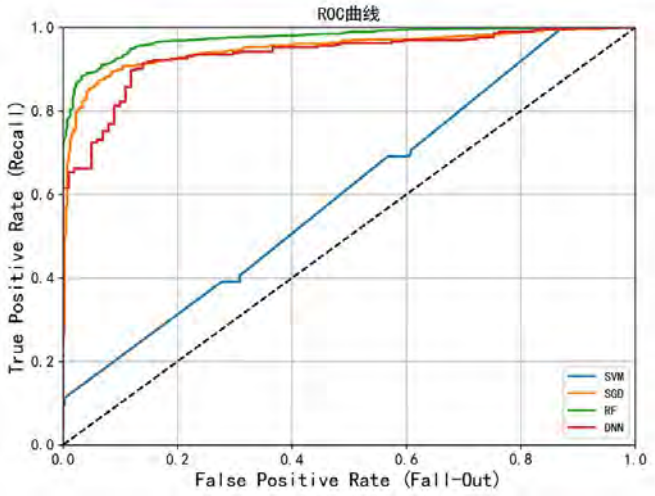


图 5-5 化合物 CYP3A4 在四种预测模型下的 ROC 曲线对比图

由图 5-5 可以观察出，随机森林 RF 的 ROC 曲线最靠左，AUC 最大，预测效果最好。而支持向量机 SVM 的 ROC 曲线最靠右，AUC 最小，预测效果最差；而 SGD 和 DNN 的 ROC 曲线很接近，随着时间的增加趋于一致，但是都没有 RF 的预测效果好。根据上述评价标准中的评价指标对化合物 CYP3A4 在四种模型下的 AUC 值、训练集准确率以及测试集准确率进行计算，得到结果如下表 5-2 所示，再根据四种模型的评价指标绘制出统计图进行清晰可观的对比分析，绘制的模型对比图如下图 5-6 所示：

表 5-2 化合物 CYP3A4 在四种模型下的评价指标结果

CYP3A4	AUC 值	训练集准确率	测试集准确率
SVM	0.6047	0.7530	0.7318
SGD	0.9498	0.8675	0.8304
DNN	0.9351	0.8450	0.8765
RF	**0.9752**	**0.9353**	**0.9129**

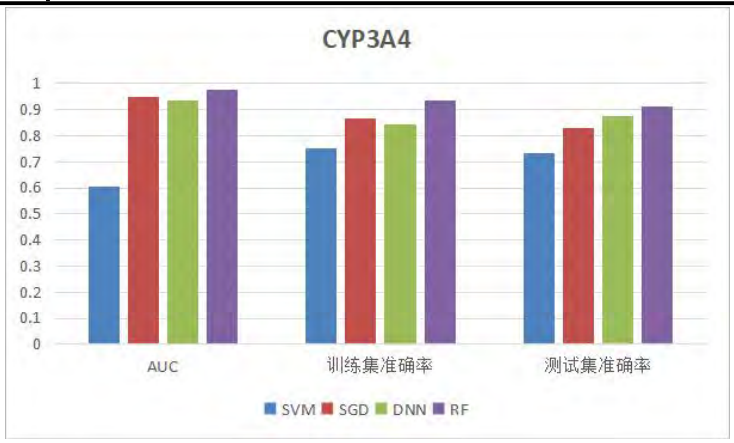


图 5-6 化合物 CYP3A4 在四种模型下的评价指标对比图

在图 5-6 中我们可以看出 RF 的 AUC 值最高，训练集准确率与测试集准确率最高，用随机森林算法构建化合物 CYP3A4 的分类预测模型最优。

5.5.3 hERG 分类模型求解

使用 python 对四种模型（SVM、SGD、DNN、RF）实现在化合物 hERG 变量下的 ROC 曲线绘制，对比图如下图 5-7 所示：

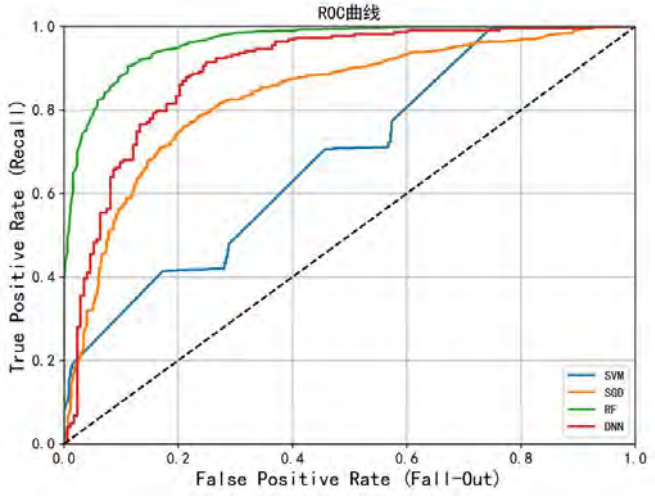


图 5-7 化合物 hERG 在四种预测模型下的 ROC 曲线对比图

由图 5-7 可以观察出，随机森林 RF 的 ROC 曲线最靠左，AUC 最大，预测效果最好。而支持向量机 SVM 的 ROC 曲线最靠右，AUC 最小，预测效果最差。根据上述评价标准中的评价指标对化合物 hERG 在四种模型下的 AUC 值、训练集准确率以及测试集准确率进行计算，得到结果如下表 5-3 所示，再根据四种模型的评价指标绘制出统计图进行清晰可观的对比分析，绘制的模型对比图如下图 5-8 所示：

表 5-3 化合物 hERG 在四种模型下的评价指标结果

hERG	AUC 值	训练集准确率	测试集准确率
SVM	0.6866	0.6162	0.5721
SGD	0.8318	0.7125	0.7369
DNN	0.8945	0.8313	0.8213
RF	**0.9623**	**0.8980**	**0.8305**

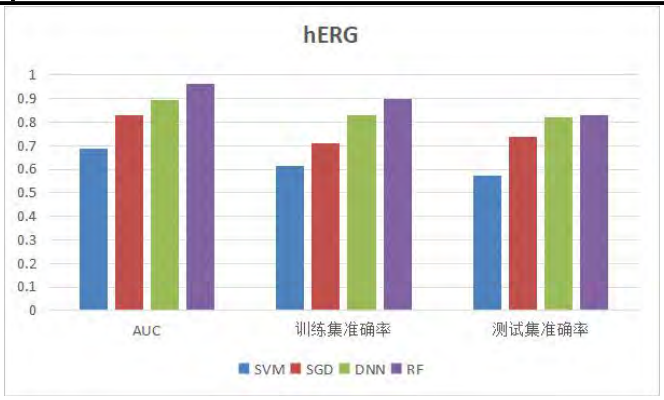


图 5-8 化合物 hERG 在四种模型下的评价指标对比图

在图 5-8 中我们可以看出 RF 的 AUC 值最高，训练集准确率与测试集准确率最高，用随机森林算法构建化合物 hERG 的分类预测模型最优。

5.5.4 HOB 分类模型求解

使用 python 对四种模型（SVM、SGD、DNN、RF）实现在化合物 HOB 变量下的 ROC 曲线绘制，对比图如下图 5-9 所示：

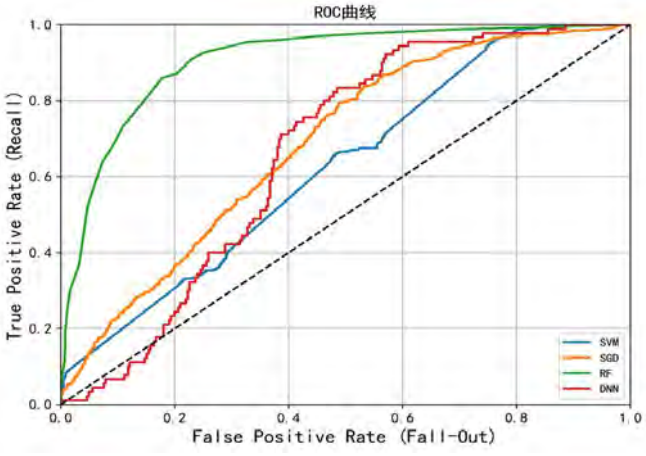


图 5-9 化合物 HOB 在四种预测模型下的 ROC 曲线对比图

由图 5-9 可以观察出，随机森林 RF 的 ROC 曲线最靠左，AUC 最大，预测效果最好。而支持向量机 SVM 的 ROC 曲线最靠右，AUC 最小，预测效果最差。根据上述评价标准中的评价指标对化合物 HOB 在四种模型下的 AUC 值、训练集准确率以及测试集准确率进行计算，得到结果如下表 5-4 所示，再根据四种模型的评价指标绘制出统计图进行清晰可观的对比分析，绘制的模型对比图如下图 5-10 所示：

表 5-4 化合物 HOB 在四种模型下的评价指标结果

HOB	AUC 值	训练集准确率	测试集准确率
SVM	0.6254	0.7613	0.7140
SGD	0.6862	0.6434	0.6450
DNN	0.6623	0.7407	0.7478
RF	**0.9069**	**0.8512**	**0.8201**

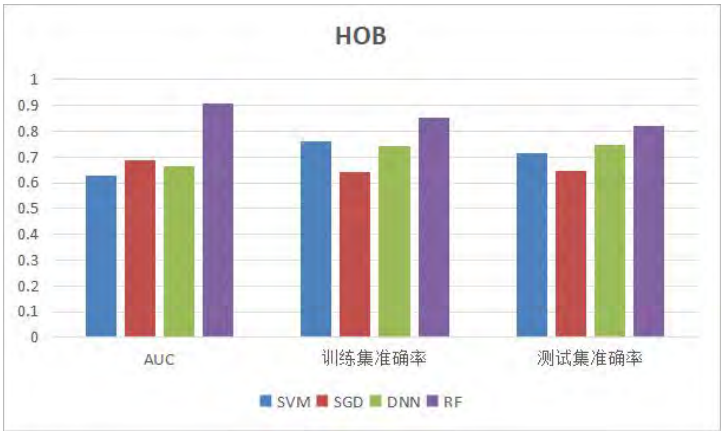


图 5-10 化合物 HOB 在四种模型下的评价指标对比图

在图 5-10 中我们可以看出 RF 的 AUC 值最高，训练集准确率与测试集准确率最高，用随机森林算法构建化合物 HOB 的分类预测模型最优。

5.5.5 MN 分类模型求解

使用 python 对四种模型（SVM、SGD、DNN、RF）实现在化合物 MN 变量下的 ROC 曲线绘制，对比图如下图 5-11 所示：

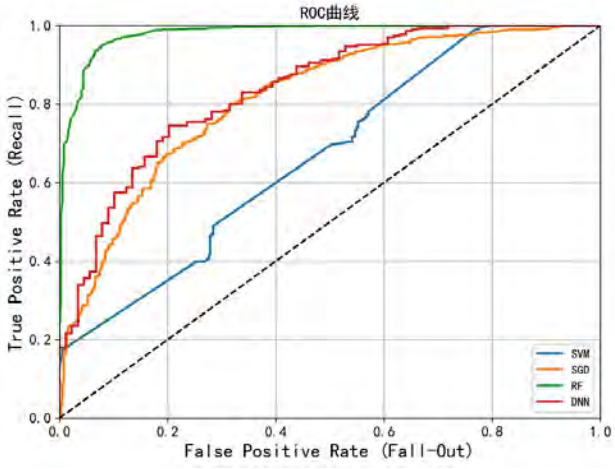


图 5-11 化合物 MN 在四种预测模型下的 ROC 曲线对比图

由图 5-11 可以观察出，随机森林 RF 的 ROC 曲线最靠左，AUC 最大，预测效果最好。而支持向量机 SVM 的 ROC 曲线最靠右，AUC 最小，预测效果最差。根据上述评价标准中的评价指标对化合物 MN 在四种模型下的 AUC 值、训练集准确率以及测试集准确率进行计算，得到结果如下表 5-5 所示，再根据四种模型的评价指标绘制出统计图进行清晰可观的对比分析，绘制的模型对比图如下图 5-12 所示：

表 5-5 化合物 MN 在四种模型下的评价指标结果

MN	AUC 值	训练集准确率	测试集准确率
SVM	0.6699	0.8158	0.7748
SGD	0.8134	0.7568	0.7591
DNN	0.8383	0.8179	0.8101
RF	**0.9803**	**0.9519**	**0.9218**

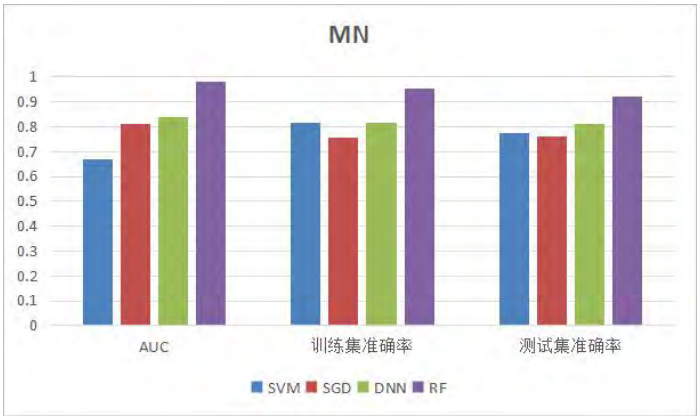


图 5-12 化合物 MN 在四种模型下的评价指标对比图

在图 5-12 中我们可以看出 RF 的 AUC 值最高，训练集准确率与测试集准确率最高，用随机森林算法构建化合物 MN 的分类预测模型最优。

5.6 模型确定及预测

通过 SVM、SGD、DNN、RF 四种模型算法分别构建对化合物 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型，在模型求解一节中我们可以清晰地观察到每种化合物在四种模型下的评价指标对比，均是 RF 算法模型的 ROC 曲线靠近左侧，且均是 RF 算法模型的 AUC 值、训练集准确率以及测试集准确率最高；由于 ROC 曲线越靠近左侧，说明 AUC 值越大，表示分类效果越好，并且 RF 算法模型的训练集准确率和测试集准确率都高于其他三种模型，因此确定用随机森林算法模型作为构建化合物的 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型。最后，用上述构建的 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，预测结果如下表 5-6 所示：

表 5-6 RF 算法模型对 50 个化合物进行的预测结果

序号	Caco-2	CYP 3A4	hERG	HOB	MN	序号	Caco-2	CYP 3A4	hERG	HOB	MN
1	0	1	1	0	1	26	1	1	1	0	0
2	0	1	1	0	1	27	0	1	1	0	0
3	0	1	1	0	1	28	0	1	1	0	1
4	0	1	1	0	1	29	0	1	1	0	1
5	0	1	1	0	1	30	0	1	1	0	1
6	0	1	1	0	1	31	0	1	1	1	1
7	0	1	1	0	1	32	0	1	1	1	1
8	0	1	1	0	1	33	0	1	1	1	1
9	0	1	1	0	1	34	0	1	1	1	1
10	0	1	1	0	1	35	0	1	1	0	1
11	0	1	1	0	1	36	0	1	0	0	1
12	0	1	1	0	1	37	0	1	0	0	1
13	0	1	1	0	1	38	0	1	1	0	0
14	0	1	1	0	1	39	0	1	1	0	1
15	0	1	1	0	1	40	0	1	1	0	1
16	0	1	1	0	1	41	0	1	1	0	1
17	0	1	1	0	1	42	0	1	1	0	1
18	0	1	0	0	1	43	0	1	1	0	1
19	0	1	0	0	1	44	0	1	1	0	1
20	0	0	0	0	1	45	0	1	1	0	1
21	0	1	1	0	1	46	0	1	1	0	1
22	0	1	1	0	1	47	0	1	1	0	1
23	1	0	1	0	0	48	0	1	1	0	1
24	1	0	1	0	0	49	0	1	1	0	1
25	1	1	1	0	0	50	0	1	1	0	0

5.7 问题小结

问题三要求针对文件“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。

本文首先对附件中的数据进行预处理，采用拉依达（3σ）准则处理数据，然后剔除特征中 0 值比例大于 90%的数据，剔除了 344 个特征，数据预处理完毕。然后介绍该小节用评价指标 ROC 和 AUC 以及准确率，并将 P/R 曲线与 ROC 曲线的优缺点进行简要对比，说明在该数据集分布不均衡的条件下，用 ROC 曲线来评价比较合适；其次在问题二模型建立的基础上新增 SGD 模型，分别用 SVM、SGD、DNN、RF 四种模型算法构建化合物的 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型，经过对比分析可以发现，在五种分类预测模型中，均是 RF 的 ROC 曲线靠近左侧，由于 ROC 曲线越靠近左侧表示分类效果越好，而且根据上述评价标准中的评价指标对化合物 Caco-2、CYP3A4、hERG、HOB、MN 在四种模型下的 AUC 值、训练集准确率以及测试集准确率进行计算并比较，观察到在五种分类预测模型中，RF 的 AUC 值最高，训练集准确率与测试集准确率最高，因此选择用随机森林算法构建化合物 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型。最后用所构建的 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列出预测结果。

6、问题四的模型建立与求解

6.1 问题分析

问题四要求寻找符合条件的分子描述符，并确定这些分子描述符的取值范围，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。本文以第二问的回归预测模型和第三问的分类模型为基础，建立目标规划模型。考虑到该目标规划要动态的结合前两问的模型，传统解法不适合用于求解本题，而现代优化算法：遗传算法适合这种问题的求解，故用遗传算法求解此问目标规划。优化以后，通过适应度值来评价其是否达到优化目标，对模型做出评价。然后，选择符合条件的优化样本，观察其决策变量（部分分子描述符）的统计特征，确定最终的分子描述符及其范围。最后，对找到的分子描述做关于 pIC50 的敏感性分析，对选出的分子描述符变化范围的进行修正。

问题四的思路流程图如下图 6-1 所示：

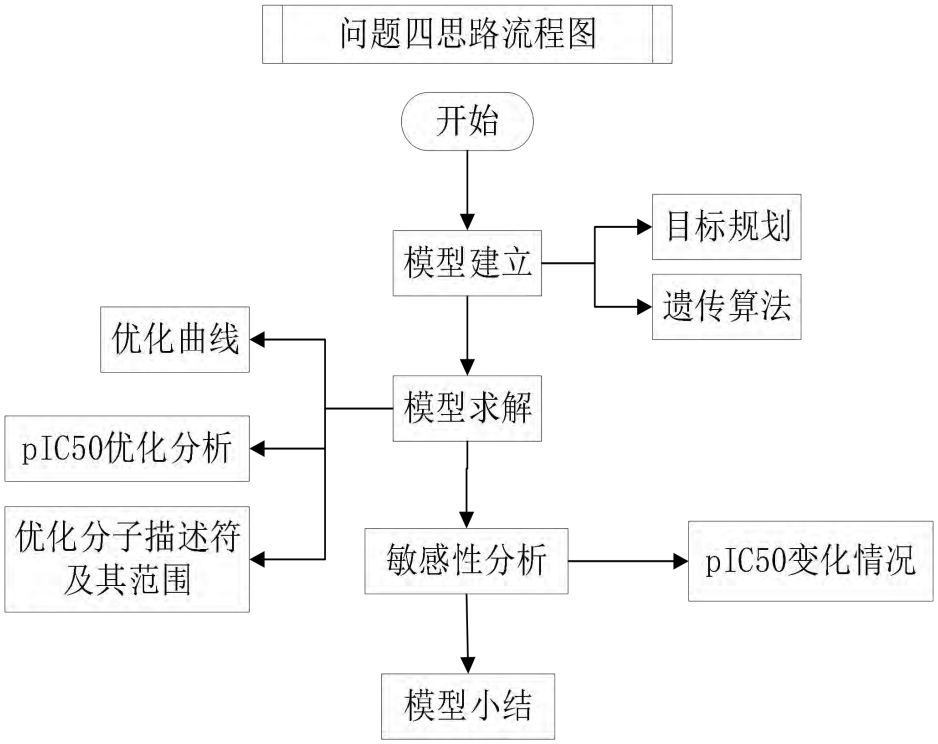


图 6-1 问题四思路流程图

6.2 模型建立

6.2.1 目标规划

目标规划是由线性规划发展演变而来，线性规划归根到底是要研究资源的有效分配和利用，模型特点是在满足一组约束条件的基础上，找出所定目标的最大值或最小值。

本文中目标规划的数学模型可表示为：

$$\begin{aligned} &max \cdot pIC50 = f(x_i, y_i) \\ &s.t. \begin{cases} x_{imin} < x_{imax}, i = 1,2,3,\ldots,20 \\ y_1 + y_2 + \overline{y_3} + y_4 + \overline{y_5} \geq 3 \end{cases} \end{aligned} \tag{8}$$

其中，x_i 表示第一问提取出的 20 个分子描述符的特征数据，y_i 代表 ADMET 性质的标签，满足题目中的要求在给定的 5 个 ADMET 性质中，至少三个性质较好的要求，优化目标是使 pIC50 的值尽可能大。

6.2.2 遗传算法

遗传算法，Genetic Algorithm，是一种解决问题的方法，它模拟大自然中种群在选择压力下的演化，从而得到一个问题的近似解。遗传算法遵循适者生存、优胜劣汰的原则，是一类借鉴生物界自然选择和自然遗传机制的随机化搜索算法。遗传算法模拟一个人工种群的进化过程，通过选择（Selection）、交叉（Crossover）以及变异（Mutation）等方法，在每次迭代中都保留一组候选个体，重复此过程，种群经过若干代进化后，理想情况下漆适应度达到近似最优的状态。遗传算法自从被提出来，得到了广泛的应用，特别是在函数优化、生产调度、模式识别、神经网络、自适应控制等领域，遗传算法发挥了很大的作用，提高了一些问题求解的效率。

遗传算法流程示意图如下图 6-2 所示：

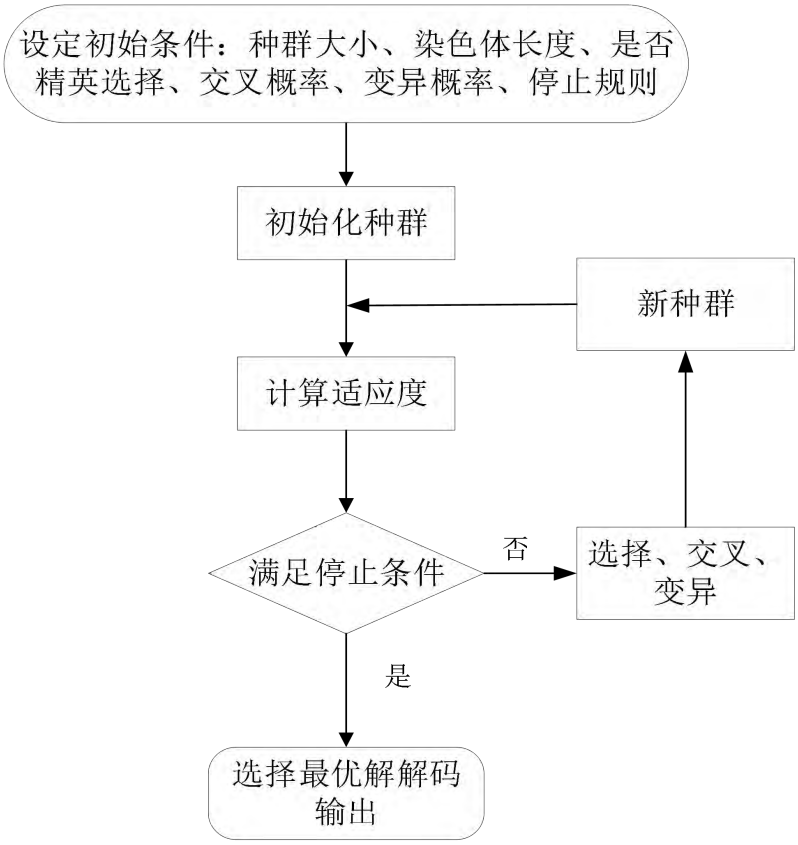


图 6-2 遗传算法流程示意图

问题四基于遗传算法的优化模型构建流程为：

1、首先采用二进制编码的方式对输入分子描述符数据 x_i 进行编码，随机产生基于二进制编码的初始化种群，定义适应度函数和终止条件；

2、计算初始化种群中的每个个体的适应度函数值；

3、判断当前种群迭代次数下是否满足终止条件，若满足，则解码输出当前最优解，否则采用交叉、变异的方式对种群进行下一轮的迭代优化，采用轮盘赌法在当前种群的基础上选择进化后的种群，直到种群满足优化的终止条件，停止对分子描述符数据的优化过程，并解码输出近似最优解。

6.3 模型求解

6.3.1 优化曲线

为了可视化已建立的基于遗传算法的优化模型的优化过程，我们随机选取了附件二训练表格中的 10 个样本来表示模型对这 10 个样本数据的优化过程，样本具体的优化过程如下图 6-3 所示：

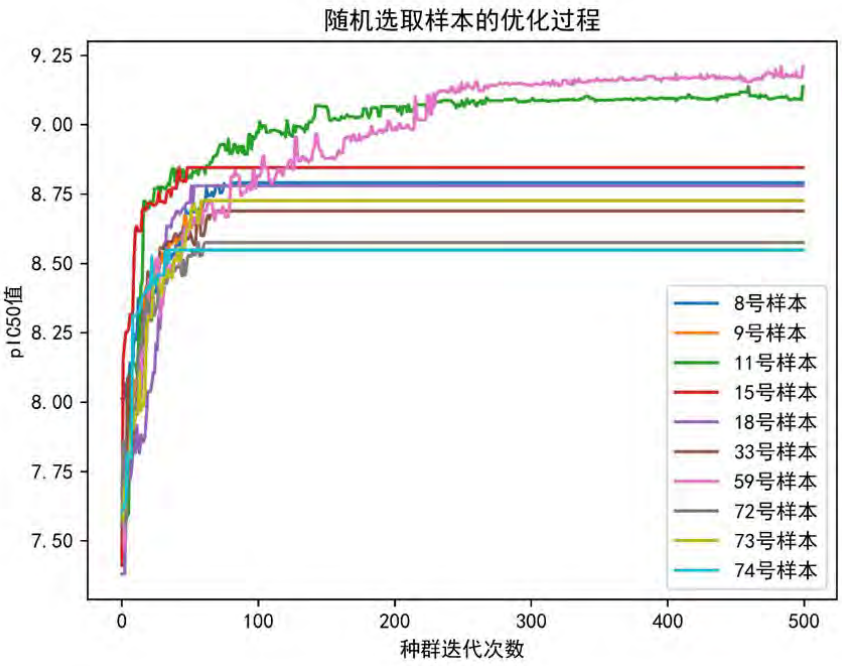


图 6-3 随机选取样本的优化过程

从图中可以看出，随着种群迭代次数的不断增大，该样本的 pIC50 预测值也在逐渐增大，其中 pIC50 的预测值由第二问我们建立的基于随机森林的回归预测模型得到，在每一个种群迭代次数下，我们都选择种群中的最优解来作为基于随机森林的预测模型的输入，从图中可以看出，11 号样本和 59 号样本随着迭代次数的增加，其 pIC50 的值也在逐步增加，最后因达到最大迭代次数限制而停止优化，而其余的 8 个样本都在不同的迭代次数下因满足停止条件达到最优解之后便停止优化了，此时得到的最大值就是此次优化的最优解，由此可见，我们的模型在不断优化化合物的分子描述符，使得化合物对抑制 $ER\alpha$ 具有更好的生物活性。

6.3.2 pIC50 优化分析

统一变量 ADMET 五个化合物分类标准的正负相关性（将 hERG 和 MN 的数值取反，和其他三个化合物方向一致），然后对五个结果累计，1 到 5 数字越大表示 ADMET 性质越好。

在性质 ADMET 中，ADME 主要指化合物的药代动力学性质，描述了化合物在生物体内的浓度随时间变化的规律，T 主要指化合物可能在人体内产生的毒副作用。一个化合物的活性再好，如果其 ADMET 性质不佳，比如很难被人体吸收，或者体内代谢速度太快，或者具有某种毒性，那么其仍然难以成为药物，因而还需要进行 ADMET 性质优化。

优化前后不同 ADMET 对应的 pIC50 分布情况分别如下图 6-4 所示：

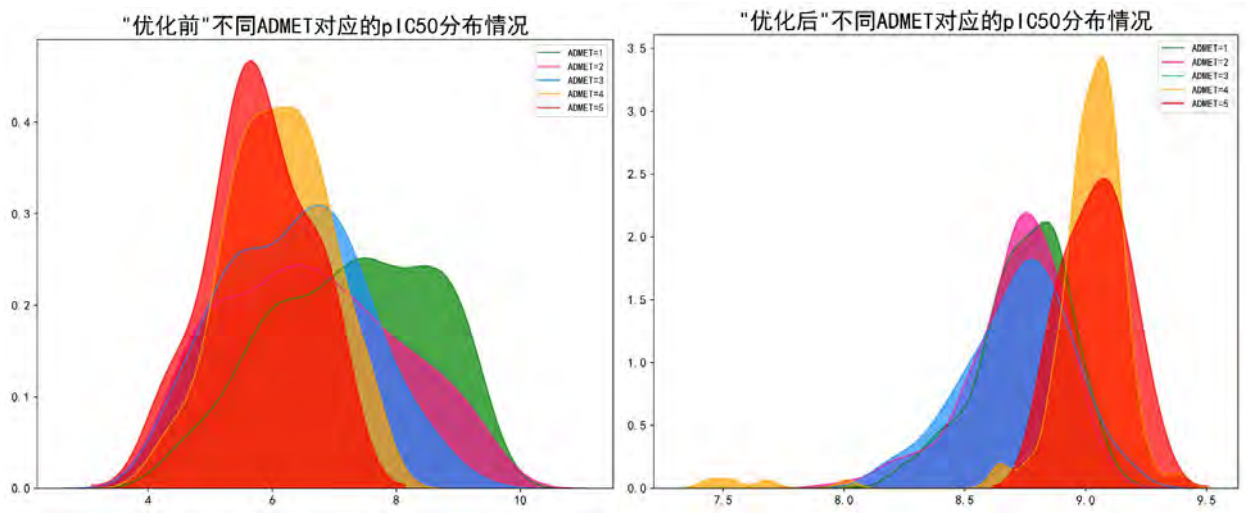


图 6-4 优化前、后不同 ADMET 对应的 pIC50 分布情况

由以上优化前后的两图示对比可知，ADMET 整体更优，pIC50 分布情况整体右移，数值变大，其峰值变大，优化效果明显。

优化前、后 pIC50 统计特征分布情况对比图如下图 6-5 所示：

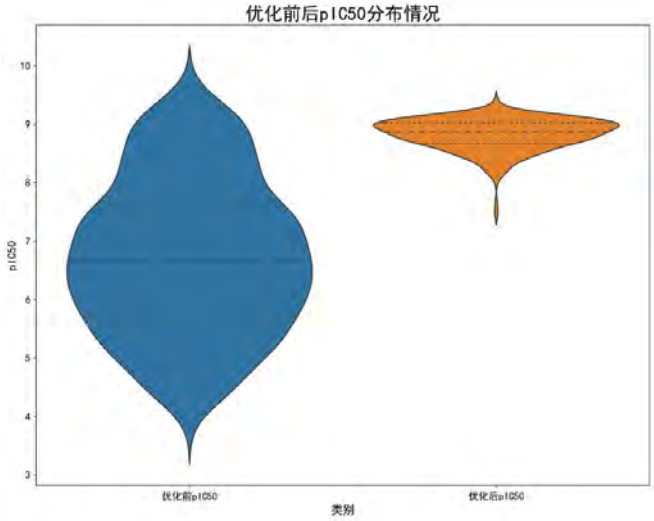


图 6-5 优化前、后 pIC50 分布情况

在上图 6-5 中，可以形象的观察到优化情况，图形的形状上下限表示最大值与最小值范围，中间的虚线表示均值，上下的虚线表示上四分位和下四分位，形状的宽度表示该数值出现的频数，优化后的值明显变大，优化效果明显。

6.3.3 优化分子描述符及其范围

通过对优化前后 pIC50 的统计特性和对应不同 ADMET 的分析易看出，本模型中选择的 20 个操作变量对 pIC50 有显著的影响。另外考虑第一问分析的这 20 个决策变量是影响 pIC50 的主要特征，故选取这 20 个决策变量作为本问要寻找的分子描述符，并将各个分子描述符[最小值，最大值]的区间作为该分子描述符的使 pIC50 和 ADMET 均达到优化的取值区间。

经过优化后的 20 个分子描述符取值范围如下表 6-1 所示：

表 6-1 优化后各分子描述符取值范围

变量	范围	变量	范围	变量	范围
SHsOH	0.428-2.248	minsOH	5.107-11.732	nHBAcc	0-10
Lipoaffinity Index	8.149-19.007	ATSc5	-0.462-0.459	gmin	-5.686-1.015
MDEC-23	0.635-53.984	WTPT-5	0.085-21.607	WTPT-4	0.172-21.865
BCUTc-1h	0.078-0.330	XLogP	-1.197-7.811	ATSp5	2375.739-7484.904
hmin	-0.566-0.381	maxssO	0.026-6.516	BCUTp-1h	10.4990-15.956
ATSc1	0.035-1.135	minHBa	-2.454-8.982	maxaaCH	1.076-2.483
ATSc3	-0.317-0.265	TopoPSA	12.470-207.734		

根据优化后分子描述符的数据，绘制出了部分主要操作变量范围分布图，如下图 6-6 所示，上下限表示最大值与最小值范围，中间的虚线表示均值，上下的虚线表示上四分位和下四分位，形状的宽度表示该数值出现的频率。分子描述符数据各自特征整体数据分布较为集中。

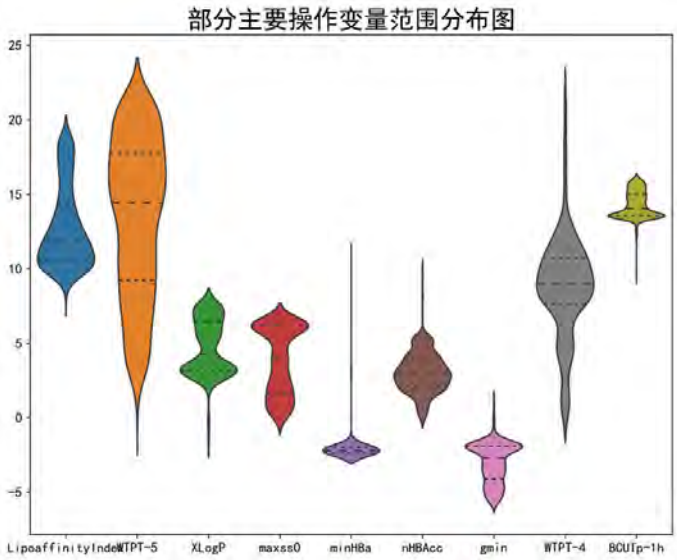
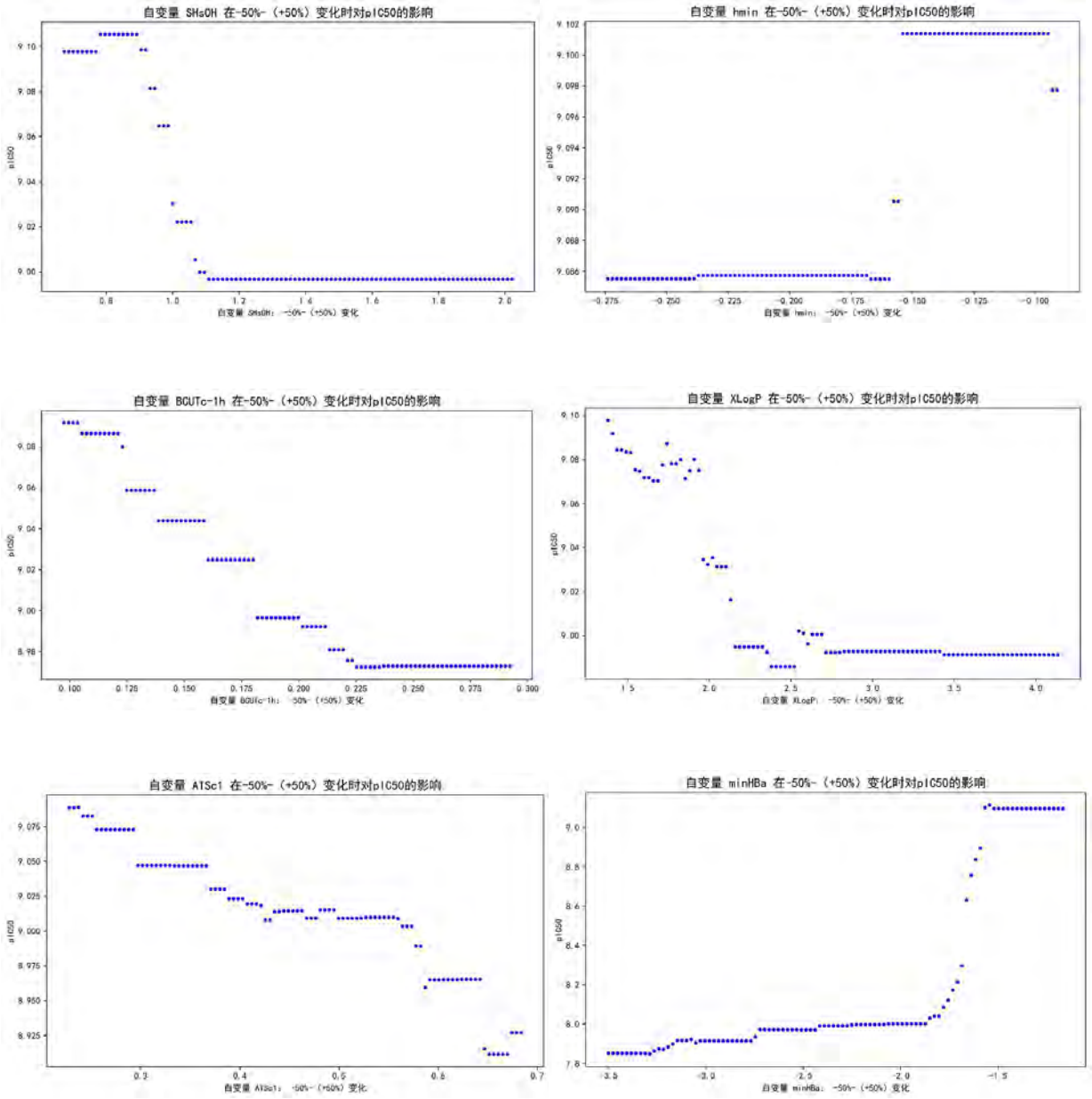


图 6-6 部分主要操作变量范围分布图

6.4 灵敏度分析

为探究建模主要变量分子描述符对生物活性的影响效果，本节考虑使用控制变量法对各个分子描述符进行灵敏性分析，在此基础之上对分子描述符的取值范围进行修正。

对于随机森林法构建的预测模型，本节优化后的选取第 11 号样本，将 20 个化合物分子描述符在-50%到+50%幅度范围内以 1%步长进行测试，从 20 个特征中随机选取一个化合物分子描述符（自变量），采用控制变量法保持其他变量不变，对其预测结果进行测试分析该变量对结果的影响，灵敏度分析图如下如 6-7 所示：



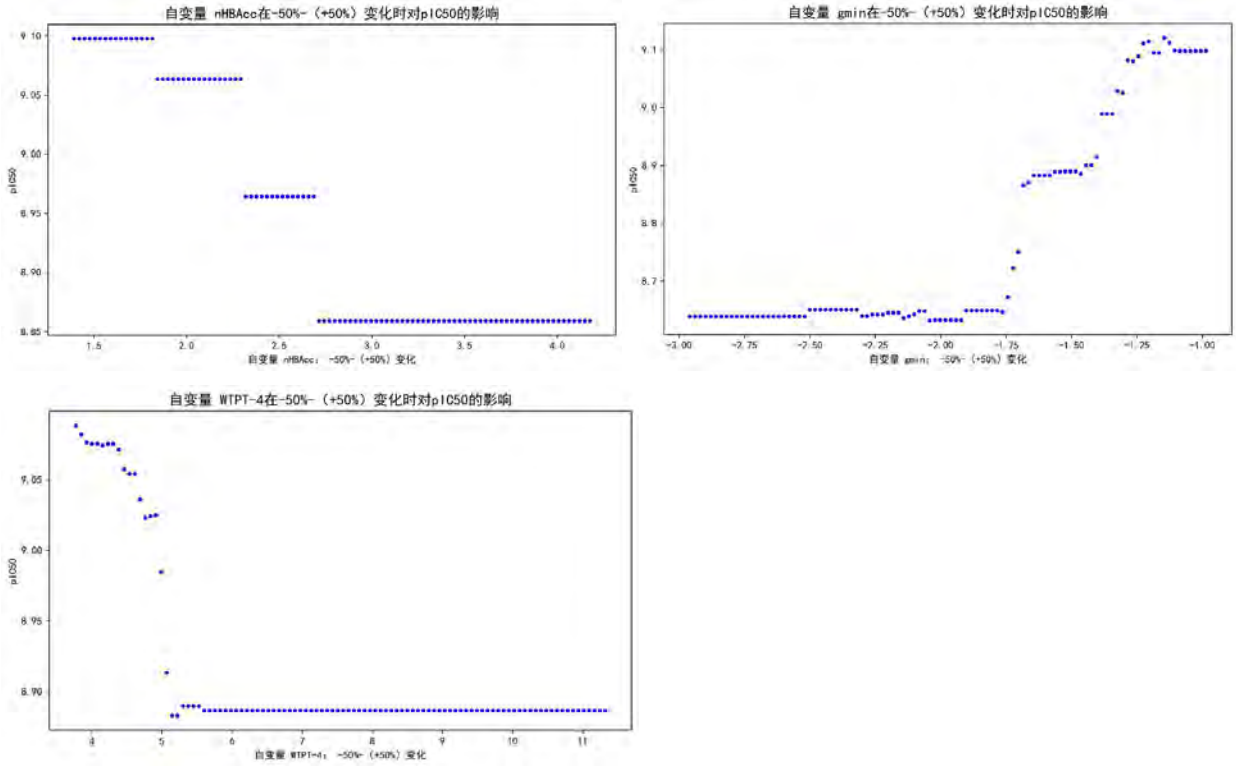


图 6-7 分子描述符对生物活性的灵敏度分析图

由上述灵敏度分析图可以看出，不同分子描述符对生物活性的灵敏度影响效果也大为不同，大致可分为先下降后稳定型、持续性下降型、先稳定后增加型三种趋势，比如变量 WTPT-4 在误差允许的范围内在前 20%的变化波动下下降速度比较快而后保持稳定；变量 ATSc1 在前 25%的变化波动下保持持续缓慢下降；而变量 gmin 在 1%为步长的变化波动下呈现先稳定而后逐步增加的趋势。

基于各个分子描述符对 pIC50 的影响，对部分分子描述符的变化范围进行微调，结果如下表 6-2：

表 6-2 对优化后的分子描述符取值范围进行修正

变量	范围	变量	范围	变量	范围
SHsOH	0.428-1	minsOH	5.107-11.732	nHBAcc	0-2.5
Lipoaffinity Index	8.149-19.007	ATSc5	-0.462-0.459	gmin	-1.75 -1.015
MDEC-23	0.635-53.984	WTPT-5	0.085-21.607	WTPT-4	0.172-5
BCUTc-1h	0.078-0.175	XLogP	-1.197-2	ATSp5	2375.739-7484.904
hmin	-0.15-0.381	maxssO	0.026-6.516	BCUTp-1h	10.4990-15.956
ATSc1	0.035-0.6	minHBa	-1-8.982	maxaaCH	1.076-2.483
ATSc3	-0.317-0.265	TopoPSA	12.470-207.734		

6.5 问题小结

问题四要求建立化合物活性优化模型，来选取适合的分子描述符及其范围。本节使用第二问 pIC50 回归模型和第三问的 ADMET 分类模型对此问进行求解。以 pIC50 为优化目标，ADMET 最少三个性质较好为约束条件，pIC50 回归模型中的 20 个变量作为决策变量，建立目标规划模型。然后，使用遗传算法对该目标规划模型进行求解。在求解过程中，画出来了优化迭代曲线，并对比了优化后和优化前的 pIC50 分布，可以清晰地看出优化后 pIC50 平均值明显增大，波动变小。由此可得出 20 个决策变量对优化模型有较强的优化效果，把这 20 个变量作为要寻找的分子描述符变量，并取它们的最大-最小值作为优化取值范围。最后，对 20 个分子描述符进行敏感性分析，采取控制变量法观察自变量在-50%-50%范围内以 1%步长浮动时对因变量的影响，来进一步修正 20 个分子描述符的取值范围。

7、模型总结

7.1 模型的评价

优点：

(1) 本文结合附件所给数据集，对数据进行了充分的预处理，并利用多种算法模型对结果进行对比分析，得出此模型的评价指标最优，准确率最高。

(2) 随机森林算法模型本身精度比大多数单个算法模型好，准确性比较高；能够处理高纬度的数据，对数据集的适应能力强，训练速度快。

(3) 利用遗传算法进行优化，可以快速将全体解搜索出来，不会陷入局部最优解的陷阱中。

缺点：

(1) 由于数据集大小具有一定的局限性，在处理问题时可能会存在一些误差。

(2) 当随机森林模型中的决策树个数很多时，训练所需的空间和时间会比较大。

(3) 在某些噪音比较大的样本集和数据集中，RF 模型容易陷入过拟合。

7.2 模型的改进

1、问题二中，支持向量回归、随机森林、神经网络和梯度提升四种算法模型的拟合度整体不高；

2、问题三的分类准确率在 95%以下，一方面是受制于数据样本数据比较少，另一方面是没有进行数据归一化，然而归一化数据对于问题四的变量范围的求解会很不方便，如果有条件可以收集更多数据，在时间允许的情况下对对数据做归一化处理；

3、在问题四中，用 python 编译程序 100 个数据集观察时间 30 分钟，在时间条件允许的情况下可以把 1975 个数据都编译完。

参考文献

[1] 张德丰，MATLAB R2015b 数学建模，北京：清华大学出版社，1-388，2016.
[2] 余胜威，MATLAB 数学建模经典案例实战，北京：清华大学出版社，158-574，2014.
[3] 木仁，吴建军，李娜，MATLAB 与数学建模，北京：科学出版社，1-261，2018.
[4] 奥雷利安·杰龙著，宋能辉，李娴译，机器学习实战：基于 Scikit-Learn、Keras 和 Tensorflow，北京：机械工业出版社，1-569，2020.
[5] 司守奎，孙兆亮，数学建模算法与应用，北京：国防工业出版社，1-471，2015.
[6] 郭君兰，王丽华，胡彦伟，焦智民.雌激素受体β亚型在乳腺癌组织中的表达及与临床病理特征的关系，中国现代医学杂志。
[7] 贺红,朱慧,杨发林,王翠萍,许伟华.雷洛昔芬对血管内皮细胞 ERE 转录活性的影响及与雌激素受体亚型的关系[J].山东大学学报(医学版),2004(05):555-557.
[8] Bai Jiawang,Li Yiming,Li Jiawei,Yang Xue,Jiang Yong,Xia Shu-Tao. Multinomial random forest[J]. Pattern Recognition,2022,122.
[9] Astuti Suryani Dyah,Tamimi Mohammad H.,Pradhana Anak A.S.,Alamsyah Kartika A.,Purnobasuki Hery,Khasanah Miratul,Susilo Yunus,Triyana Kuwat,Kashif Muhammad,Syahrom Ardiyansyah. Gas sensor array to classify the chicken meat with E. coli contaminant by using random forest and support vector machine[J]. Biosensors and Bioelectronics: X,2021,9.
[10] Zare Alaa,Postovit Lynne Marie,Githaka John Maringa. Robust inflammatory breast cancer gene signature using nonparametric random forest analysis[J]. Breast Cancer Research,2021,23(1).

附录

Python 和部分 MATLAB 代码

问题一代码：##

#######Python#######

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedShuffleSplit
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde
from matplotlib.colors import LogNorm
import matplotlib.image as mpimg
from sklearn import preprocessing
from sklearn.ensemble import RandomForestClassifier
os.chdir(r'C:\Users\jupyter-notebook\数学建模\2021\第一问')
#加载自变量数据集
Molecular_Descriptor=pd.read_excel('MD_train_dropzero0.5_mval.xlsx',header=0,encoding='gb
2312')
Molecular_Descriptor=Molecular_Descriptor.iloc[:,1:]
print(Molecular_Descriptor.shape)
Molecular_Descriptor.head()
#加载因变量数据集
ERa_activity= pd.read_excel('ERα_activity.xlsx',header = 0,encoding='gb2312')
ERa_activity=ERa_activity.iloc[:,2]
print(ERa_activity.shape)
ERa_activity.head()
#随机森林降维
rnd_clf = RandomForestClassifier(n_estimators=500, n_jobs=-1, random_state=42)
rnd_clf.fit(Molecular_Descriptor, ERa_activity.astype('int'))
rnd_clf.feature_importances_.shape
import matplotlib as mpl
# 为了显示中文
mpl.rcParams['font.sans-serif'] = [u'SimHei']
mpl.rcParams['axes.unicode_minus'] = False
#画图
plt.figure(figsize=(10,5.5))
x_data = rnd_clf.feature_importances_
plt.bar(x = range(0,len(x_data)),height = x_data,align='center',color='b')
plt.xlabel('特征',labelpad = 19)   # 控制标签和坐标轴的距离
```

```
plt.ylabel('贡献率',labelpad =10)
plt.title('随机森林测得各个特征的贡献率', fontsize=22)
plt.savefig('随机森林测得各个特征的贡献率.png', dpi=500, bbox_inches='tight') # 解决图片不清晰，不完整的问题
# plt.title('不同地区的的订单金额',pad=15)
plt.show()
#排序后贡献率图
x = np.arange(0,len(feature_importances_descend))
plt.figure(figsize=(12,9), dpi= 80)
plt.title('排序后每个特征贡献率百分比', fontsize=22)
plt.xlabel('特征',labelpad = 19)  # 控制标签和坐标轴的距离
plt.ylabel('贡献率',labelpad =10)
plt.plot(x,feature_importances_descend)
plt.savefig('随机森林排序后每个特征贡献率百分比.png', dpi=500, bbox_inches='tight') # 解决图片不清晰，不完整的问题
plt.show()
#30 个特征相关性图
Molecular_Descriptor_30=Molecular_Descriptor.iloc[:,index_id]
print(Molecular_Descriptor_30.shape)
import seaborn as sns
# 显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
# Plot
plt.figure(figsize=(17,14), dpi= 80)
sns.heatmap(Molecular_Descriptor_30.corr(),
xticklabels=Molecular_Descriptor_30.corr().columns,
yticklabels=Molecular_Descriptor_30.corr().columns, cmap='RdYlGn', center=0, annot=True)
# Decorations
plt.title('随机森林前 30 个特征之间的相关性', fontsize=22)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.savefig('随机森林前 30 个特征之间的相关性.png', dpi=500, bbox_inches='tight')
plt.show()
#20 个特征相关性图
Molecular_Descriptor_20=Molecular_Descriptor.iloc[:,[99,144,211,29, 142, 12, 14 ,123, 16,240, 243, 138, 112, 232, 193, 143, 239, 26, 31, 134]]
print(Molecular_Descriptor_20.shape)
import seaborn as sns
# 显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

```
# Plot
plt.figure(figsize=(17,14), dpi= 80)
sns.heatmap(Molecular_Descriptor_20.corr(),
xticklabels=Molecular_Descriptor_20.corr().columns,
yticklabels=Molecular_Descriptor_20.corr().columns, cmap='RdYlGn', center=0, annot=True)
plt.savefig('筛选的 20 个特征.png', dpi=500, bbox_inches='tight') # 解决图片不清晰，不完整的问题
# Decorations
plt.title('筛选的 20 个特征', fontsize=22)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.savefig('筛选的 20 个特征.png', dpi=500, bbox_inches='tight') # 解决图片不清晰，不完整的问题
plt.show()
#######MATLAB#######
clc
clear
close
x=xlsread('MD_train_dropzero0.5_mval.xlsx','C2:IM1975');
[n,m]=size(x);
%% 数据的归一化处理
[X,ps]=mapminmax(x');
ps.ymin=0.002; % 归一化后的最小值
ps.ymax=0.996; % 归一化后的最大值
ps.yrange=ps.ymax-ps.ymin; % 归一化后的极差,若不调整该值, 则逆运算会出错
X=mapminmax(x',ps);
X=X';
%% 计算第 j 个指标下，第 i 个记录占该指标的比重 p(i,j)
for i=1:n
    for j=1:m
        p(i,j)=X(i,j)/(sum(X(:,j)));
    end
end
%% 计算第 j 个指标的熵值 e(j)
k=1/log(n);
for j=1:m
    e(j)=-k*sum(p(:,j).*log(p(:,j)));
end
d=ones(1,m)-e;  % 计算信息熵冗余度
w=d./sum(d);    % 求权值 w
s=w*p';         % 求综合得分
%% 排序
```

```
[gsort,ind]=sort(w,'descend')
plot(gsort)
title('权值-特征图')
xlabel('特征')
ylabel('权值')
```
问题二代码：
```
#########################################################
#######Python#######
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedShuffleSplit
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde
from matplotlib.colors import LogNorm
from sklearn.model_selection import train_test_split
import matplotlib.image as mpimg
from sklearn import preprocessing
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
os.chdir(r'C:\Users\jupyter-notebook\数学建模\2021\第二问')
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)
# 为了显示中文
mpl.rcParams['font.sans-serif'] = [u'SimHei']
mpl.rcParams['axes.unicode_minus'] = False
import numpy as np
from math import sqrt

def accuracy_score(y_true, y_predict):
    """计算 y_true 和 y_predict 之间的准确率"""
    assert y_true.shape[0] == y_predict.shape[0], \
        "the size of y_true must be equal to the size of y_predict"
    return sum(y_true == y_predict) / len(y_true)
# MSE：均方误差
def mean_squared_error(y_true, y_predict):
```

```
    """计算 y_true 与 y_predict 之阿的 MSE"""
    assert len(y_true) == len(y_predict), \
        "the size of y_true must be equal to the size of y_predict"
    return np.sum((y_true - y_predict) ** 2) / len(y_true)
# RMSE：均方根误差
def root_mean_squared_error(y_true, y_predict):
    """计算 y_true 与 y_predict 之阿的 RMSE"""
    return sqrt(mean_squared_error(y_true, y_predict))
# MAE：平均绝对误差
def mean_absolute_error(y_true, y_predict):
    """计算 y_true 与 y_predict 之阿的 MAE"""
    assert len(y_true) == len(y_predict), \
        "the size of y_true must be equal to the size of y_predict"
    return np.sum(np.absolute(y_true - y_predict)) / len(y_true)
# R Square
def r2_score(y_true, y_predict):
    """计算 y_true 和 y_predict 之间的 R Square"""
    return 1 - mean_squared_error(y_true, y_predict) / np.var(y_true)
#加载数据
y= pd.read_excel('ERα_activity.xlsx',header = 0,encoding='gb2312')
y=y.iloc[:,2]
##SVR
import time
start =time.clock()
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=.2,random_state=49)
from sklearn.svm import SVR
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import reciprocal, uniform
svm_clf = SVR(kernel='rbf')
param_distributions = {"gamma": reciprocal(0.001, 0.01), "C": uniform(1, 10)}
rnd_search_cv = RandomizedSearchCV(svm_clf, param_distributions, n_iter=100,verbose=2,
cv=5,scoring='r2', random_state=42,n_jobs=1)
rnd_search_cv.fit(X_train, y_train)
end = time.clock()
print('Running time: %s Seconds'%(end-start))
# 搜寻到的最佳模型
rnd_search_cv.best_estimator_
# 进行模型性能估计
y_pred1 = rnd_search_cv.best_estimator_.predict(X_train)
y_pred2 = rnd_search_cv.best_estimator_.predict(X_val)
print("训练 RMSE:",root_mean_squared_error(y_train, y_pred1))
print("训练 MAE:",mean_absolute_error(y_train, y_pred1))
```

```
print("得分:",np.sqrt(rnd_search_cv.score(X_val,y_val)))
print("测试 RMSE:",root_mean_squared_error(y_val, y_pred2))
print("测试 MAE:",mean_absolute_error(y_val, y_pred2))
print('拟合度：',r2_score(y_val,y_pred2))
#拟合效果图
aa_true=pd.DataFrame(y_val[:30]).values
aa_val=y_pred2[:30]
plt.figure(figsize=(10,5.5))
plt.plot(aa_true, "b.-",label="ture")
plt.plot(aa_val, "r.-",label="val")
plt.xlabel("30 个数据编号")
plt.ylabel("pIC50")
plt.legend(['test_true','test_pre'])
plt.title("SVR 30 个测试数据真实值和预测值", fontsize=14)
plt.savefig('SVR 30 个测试数据真实值和预测值.png', dpi=500, bbox_inches='tight') # 解决图片不清晰，不完整的问题
##RF
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=.2,random_state=49)
param_distribs = {
        'n_estimators': randint(low=1, high=200),
        'max_features': randint(low=1, high=8),
    }
forest_reg = RandomForestRegressor(random_state=42)
rnd_search = RandomizedSearchCV(forest_reg, param_distributions=param_distribs,
                                n_iter=100, cv=5,scoring='r2', random_state=42)
rnd_search.fit(X_train, y_train)
# 搜寻到的最佳模型
print(rnd_search.best_estimator_)
# 进行模型性能估计
y_pred1 = rnd_search.best_estimator_.predict(X_train)
y_pred2 = rnd_search.best_estimator_.predict(X_val)
print("训练 RMSE:",root_mean_squared_error(y_train, y_pred1))
print("训练 MAE:",mean_absolute_error(y_train, y_pred1))
print("得分:",np.sqrt(-rnd_search.score(X_val,y_val)))
print("测试 RMSE:",root_mean_squared_error(y_val, y_pred2))
print("测试 MAE:",mean_absolute_error(y_val, y_pred2))
print('拟合度：',r2_score(y_val,y_pred2))
#拟合图
aa_true=pd.DataFrame(y_val[:30]).values
aa_val=y_pred2[:30]
```

```
plt.figure(figsize=(10,5.5))
plt.plot(aa_true, "b.-",label="ture")
plt.plot(aa_val, "r.-",label="val")
plt.xlabel("30 个数据编号")
plt.ylabel("pIC50")
plt.legend(['test_true','test_pre'])
plt.title("RF 30 个测试数据真实值和预测值", fontsize=14)
plt.savefig('RF 30 个测试数据真实值和预测值.png', dpi=500, bbox_inches='tight') # 解决图片不清晰，不完整的问题
##GBRT
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import GradientBoostingRegressor
from scipy.stats import randint
training_scores=[]
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=.2,random_state=49)
param_distribs = {
        'max_depth': randint(low=2, high=20),
        'n_estimators': randint(low=20, high=200),
    }
gbrt = GradientBoostingRegressor(random_state=42,learning_rate=0.1)
rnd_search = RandomizedSearchCV(gbrt, param_distributions=param_distribs,
                        n_iter=100, cv=5, scoring='r2', random_state=42)
rnd_search.fit(X_train, y_train)
# 搜寻到的最佳模型
print(rnd_search.best_estimator_)
# 进行模型性能估计
y_pred1 = rnd_search.best_estimator_.predict(X_train)
y_pred2 = rnd_search.best_estimator_.predict(X_val)
print("训练 RMSE:",root_mean_squared_error(y_train, y_pred1))
print("训练 MAE:",mean_absolute_error(y_train, y_pred1))
print("得分:",np.sqrt(-rnd_search.score(X_val,y_val)))
print("测试 RMSE:",root_mean_squared_error(y_val, y_pred2))
print("测试 MAE:",mean_absolute_error(y_val, y_pred2))
print('拟合度：',r2_score(y_val,y_pred2))
#拟合图
aa_true=pd.DataFrame(y_val[:30]).values
aa_val=y_pred2[:30]
plt.figure(figsize=(10,5.5))
plt.plot(aa_true, "b.-",label='test_true')
plt.plot(aa_val, "r.-",label='test_pre')
```

```
plt.xlabel("30 个数据编号")
plt.ylabel("pIC50")
plt.legend(['test_true','test_pre'])
plt.title("GBRT 30 个测试数据真实值和预测值", fontsize=14)
plt.savefig('GBRT 30 个测试数据真实值和预测值.png', dpi=500, bbox_inches='tight') # 解决图片不清晰，不完整的问题
#用 RF 预测
X_test= pd.read_excel('MD_test2.xlsx' , header = 0 , encoding='gb2312')
print(X_test.shape)
X_test.head()
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=.2,random_state=49)
Rtree_clf = RandomForestRegressor(max_features=6, n_estimators=185,random_state=42)
Rtree_clf.fit(X_train, y_train)
# 进行模型性能估计
y_pred1 = Rtree_clf.predict(X_train)
y_pred2 = Rtree_clf.predict(X_val)
print("训练 RMSE:",root_mean_squared_error(y_train, y_pred1))
print("训练 MAE:",mean_absolute_error(y_train, y_pred1))
print("得分:",np.sqrt(-Rtree_clf.score(X_val,y_val)))
print("测试 RMSE:",root_mean_squared_error(y_val, y_pred2))
print("测试 MAE:",mean_absolute_error(y_val, y_pred2))
print('拟合度：',r2_score(y_val,y_pred2))
#预测 50 组数据并保存
y_test_pre_pIC50= Rtree_clf.predict(X_test)
y_test_pre_pIC50=pd.DataFrame(y_test_pre_pIC50)
y_test_pre_pIC50.to_excel('y_test_pre_pIC50.xlsx', index=False)
```

问题三代码：

```
##########################################################
####Python#####
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedShuffleSplit
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde
from matplotlib.colors import LogNorm
import matplotlib.image as mpimg
from sklearn import preprocessing
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)
# 为了显示中文
mpl.rcParams['font.sans-serif'] = [u'SimHei']
mpl.rcParams['axes.unicode_minus'] = False
os.chdir(r'C:\Users\jupyter-notebook\数学建模\2021\第三问')
#加载数据集
X= pd.read_excel('MD_train_dropzero0.9_mval.xlsx',header = 0,encoding='gb2312')
X=X.iloc[:,1:]
y= pd.read_excel('ADMET.xlsx',header = 0,encoding='gb2312')
y1,y2,y3,y4,y5=y.iloc[:,1],y.iloc[:,2],y.iloc[:,3],y.iloc[:,4],y.iloc[:,5]
## 数据处理
X1_train, X1_val, y1_train, y1_val = train_test_split(X, y5, test_size=.2,random_state=49)
def plot_roc_curve(fpr, tpr, label=None):
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--') # dashed diagonal
    plt.axis([0, 1, 0, 1])                                    # Not shown in the book
    plt.xlabel('False Positive Rate (Fall-Out)', fontsize=16) # Not shown
    plt.ylabel('True Positive Rate (Recall)', fontsize=16)    # Not shown
    plt.grid(True)
#SGD
#训练模型
sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42,n_jobs=200)
sgd_clf.fit(X1_train, y1_train)
y1_scores_sgd=cross_val_predict(sgd_clf,X1_train,y1_train, cv=3,method="decision_function")
fpr_sgd,tpr_sgd,thresholds_sgd=roc_curve(y1_train,y1_scores_sgd)
from sklearn.model_selection import cross_val_score
accuracy_SGD_train=sum(cross_val_score(sgd_clf,X1_train,y1_train,cv=10,scoring="accuracy")
)/10
accuracy_SGD_val=sum(cross_val_score(sgd_clf,X1_val,y1_val,cv=10,scoring="accuracy"))/1
```

```
#SVM
from sklearn.model_selection import RandomizedSearchCV
from sklearn.svm import SVC, LinearSVC
from scipy.stats import reciprocal, uniform
svm_clf = SVC(decision_function_shape="ovr", gamma="auto")
param_distributions = {"gamma": reciprocal(0.001, 0.1), "C": uniform(1, 10)}
rnd_search_cv = RandomizedSearchCV(svm_clf, param_distributions, n_iter=10, verbose=2, cv=3,n_jobs=1)
rnd_search_cv.fit(X1_train, y1_train)
# 搜寻到的最佳模型
print('搜寻到的最佳模型:',rnd_search_cv.best_estimator_)
# 最佳参数
print('最佳参数:',rnd_search_cv.best_score_)
#交叉验证
from sklearn.model_selection import cross_val_score
accuracy_SVM_train=sum(cross_val_score(rnd_search_cv.best_estimator_, X1_train, y1_train, cv=10, scoring="accuracy"))/10
accuracy_SVM_val=sum(cross_val_score(rnd_search_cv.best_estimator_, X1_val, y1_val, cv=10, scoring="accuracy"))/10
y1_scores_svm = cross_val_predict(rnd_search_cv.best_estimator_, X1_train, y1_train, cv=3,method="decision_function")
fpr_svm, tpr_svm, thresholds_svm = roc_curve(y1_train, y1_scores_svm)
#RF
from sklearn.model_selection import RandomizedSearchCV
from sklearn.svm import SVC, LinearSVC
from scipy.stats import reciprocal, uniform
from scipy.stats import randint
param_distributions = {
        'n_estimators': randint(low=1, high=200),
        'max_features': randint(low=1, high=8),
    }
forest_clf = RandomForestClassifier(random_state=42)
rnd_search_cv_forest = RandomizedSearchCV(forest_clf, param_distributions, n_iter=100, verbose=2, cv=3,n_jobs=16)
rnd_search_cv_forest.fit(X1_train, y1_train)
## 保存模型
import joblib
#保存模型
forest_clf_y5=rnd_search_cv_forest.best_estimator_
joblib.dump(forest_clf_y5,'forest_clf_y5.pkl') #将 clf 存入.pkl 的文件中
# 搜寻到的最佳模型
```

```
print('搜寻到的最佳模型:',rnd_search_cv_forest.best_estimator_)
# 最佳参数
print('最佳参数:',rnd_search_cv_forest.best_score_)
#交叉验证计算准确率
from sklearn.model_selection import cross_val_score
accuracy_RF_train=sum(cross_val_score(rnd_search_cv_forest.best_estimator_,X1_train,
y1_train, cv=10, scoring="accuracy"))/10
accuracy_RF_val=sum(cross_val_score(rnd_search_cv_forest.best_estimator_, X1_val, y1_val,
cv=10, scoring="accuracy"))/10
cross_val_score(rnd_search_cv_forest.best_estimator_,X1_train,y1_train,cv=10,scoring="accura
cy")
from sklearn.ensemble import RandomForestClassifier
# forest_clf = RandomForestClassifier(n_estimators=100, random_state=42)
y_probas_forest = cross_val_predict(rnd_search_cv_forest.best_estimator_, X1_train, y1_train,
cv=3, method="predict_proba")
y_scores_forest = y_probas_forest[:, 1] # score = proba of positive class
fpr_forest, tpr_forest, thresholds_forest = roc_curve(y1_train,y_scores_forest)
#ROC 曲线、AUC 值和准确率
plt.figure(figsize=(8, 6))                                    # Not shown
plot_roc_curve(fpr_svm, tpr_svm,'SVM')
plot_roc_curve(fpr_sgd, tpr_sgd,'SGD')
plot_roc_curve(fpr_forest, tpr_forest,'RF')
plt.legend(loc='lower right')
plt.title("ROC 曲线", fontsize=14)
# plt.legend(['SVM','SGD','RF'])
plt.show()
print('SGD 的 AUC 值：',roc_auc_score(y1_train, y1_scores_sgd))
print('SVM 的 AUC 值：',roc_auc_score(y1_train, y1_scores_svm))
print('RF 的 AUC 值：  ',roc_auc_score(y1_train, y_scores_forest))
print('SGD 交叉验证训练集准确率：    ',accuracy_SGD_train)
print('SGD 交叉验证测试集准确率：->',accuracy_SGD_val)
print('SVM 交叉验证训练集准确率：    ',accuracy_SVM_train)
print('SVM 交叉验证测试集准确率：->',accuracy_SVM_val)
print('RF 交叉验证训练集准确率：      ',accuracy_RF_train)
print('RF 交叉验证测试集准确率：  -> ',accuracy_RF_val)
#用 RF 预测
X_test= pd.read_excel('MD_test_dropzero0.9_mval.xlsx' , header = 0 , encoding='gb2312')
X_test=X_test.iloc[:,1:]
print(X_test.shape)
X_test.head()
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
```

```
# X1_train, X1_val, y1_train, y1_val = train_test_split(X, y1, test_size=.2,random_state=49)#############需要改
# forest_clf = RandomForestClassifier(n_estimators=169 ,max_features=5 ,random_state=42)
forest_clf = rnd_search_cv_forest.best_estimator_
forest_clf.fit(X1_train, y1_train)
## 模型准确率检验
y1_val_pred=forest_clf.predict(X1_val)
print('准确率：',accuracy_score(y1_val, y1_val_pred))
## 预测 test 数据
y_test_pred=forest_clf.predict(X_test)
y_test_pred=pd.DataFrame(y_test_pred)
y_test_pred.to_excel('y1_test_pred.xlsx', index=False)#############需要改
```

问题四代码：

```
##########################################################
####Python########
import joblib
import pandas as pd
import numpy as np
import time
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
class GeneticGroup:
    def __init__(self,vector_reg,vector_clf,vector_v_index,dna_size,pd_variables_range,
                 n_pop=500,crossover_rate=0.5,mutation_rate=0.1,
                 n_generations=50,model_r=np.max,model_c1=np.min,

model_c2=np.min,model_c3=np.min,model_c4=np.min,model_c5=np.min):
        self.v = vector_reg
        self.v_c = vector_clf
        self.vvi = vector_v_index
        self.m_reg = model_r
        self.m_c1 = model_c1
        self.m_c2 = model_c2
        self.m_c3 = model_c3
        self.m_c4 = model_c4
        self.m_c5 = model_c5
        self.ds = dna_size
        self.np = n_pop
        self.cr = crossover_rate
        self.mr = mutation_rate
        self.ng = n_generations
        self.pvr = pd_variables_range
        self.nv = len(self.pvr)
```

```
            self.pop = np.random.randint(2, size=(self.np, self.ds * self.nv))
            self.x = np.tile(vector_reg,self.np).reshape(self.np,-1)
            self.x_clf = np.tile(vector_clf,self.np).reshape(self.np,-1)
        #进行解码操作
        def translateDNA(self):
            pop = self.pop
            data_x = np.zeros((self.np, self.nv))
            for i in range(self.nv):
                pop_x = pop[:,i::self.nv]
                print('pop_x.shape:',pop_x.shape)
                print('nv:', self.nv)
                data_x[:,i] = pop_x.dot(2 ** np.arange(self.ds)[::-1]) / float(2 ** self.ds - 1)\
*float(self.pvr['max'][i]-self.pvr['min'][i])+float(self.pvr['min'][i])
            self.x = data_x
            self.x_clf[:,self.vvi] = self.x
            return self.x,self.x_clf   ##将二进制编码的值转换为实际的数值，这里要返回两个
值，一个是回归用的数据，一个是分类用数据
        def get_fitness(self):
            data_x,data_xclf = self.translateDNA()
            pred_reg = self.m_reg.predict(data_x)
            total_ADMET = self.m_c1.predict(data_xclf) + self.m_c2.predict(data_xclf) +
(self.m_c3.predict(data_xclf)^1) + self.m_c4.predict(data_xclf) +
(self.m_c5.predict(data_xclf)^1)
            fitness_Reg = pred_reg - np.min(pred_reg) + 1e-4
            fitness_ADMET = (total_ADMET >= 4) * 1 + 1e-4
            return fitness_Reg * fitness_ADMET
        def select(self):
            idx = np.random.choice(np.arange(self.np),size=self.np,

replace=True,p=self.get_fitness()/np.sum(self.get_fitness()))
            self.pop = self.pop[idx]
        def mutation(self, vector):
            if np.random.rand() < self.mr:
                mutate_point = np.random.randint(0,self.ds * self.nv)
                vector[mutate_point] = vector[mutate_point] ^ 1
        def crossover_and_mutation(self):
            pop = self.pop
            for i in range(self.np):
                child = self.pop[i,:]
                if np.random.rand() < self.cr:
                    mother = pop[np.random.randint(self.np)]
                    cross_points = np.random.randint(0, self.ds * self.nv)
```

```
                    child[cross_points:] = mother[cross_points:]
                self.mutation(child)
                pop[i,:] = child
            self.pop = pop
        def optimization(self):
            zy = np.zeros((self.ng,self.nv))
            for i in range(self.ng):
                self.select()
                self.crossover_and_mutation()
                fitness = self.get_fitness()
                if np.std(fitness) <= 1e-5:
                    break
                zy[i,:] = self.x[int(np.argmax(fitness)),:]
            return self.x[int(np.argmax(fitness)),:],self.x_clf[int(np.argmax(fitness)),:],zy
#加载基于随机森林的分类模型
RF_clf_Caco_2 = joblib.load('./Classify/forest_clf_Caco-2.pkl')
RF_clf_CYP3A4 = joblib.load('./Classify/forest_clf_CYP3A4.pkl')
RF_clf_hERG = joblib.load('./Classify/forest_clf_hERG.pkl')
RF_clf_HOB = joblib.load('./Classify/forest_clf_HOB.pkl')
RF_clf_MN = joblib.load('./Classify/forest_clf_MN.pkl')
#加载基于随机森林的回归预测模型
RF_reg = joblib.load('./REG/forest_reg_y.pkl')
#加载要优化的回归数据
data_reg=pd.read_excel('./Molecular_Descriptor_20.xlsx',header=0,encoding='gb2312')#(1974,2
0)
label_reg = pd.read_excel('./ERα_activity.xlsx',header = 0,encoding='gb2312').iloc[:,2]
#加载要优化的分类数据
data_clf=pd.read_excel('./MD_train_dropzero0.9_mval.xlsx',header=0,encoding='gb2312').iloc[:,
1:]
#为 0.9 时剔除 344 个特征，第一次剔除了 26 个特征(1974,359)
label_clf= pd.read_excel('./ADMET.xlsx',header = 0,encoding='gb2312')
Caco_2,CYP3A4,hERG,HOB,MN=label_clf.iloc[:,1],label_clf.iloc[:,2],label_clf.iloc[:,3],label_c
lf.iloc[:,4],label_clf.iloc[:,5]
print('shape of datareg:{}\tshape of data_clf:{}'.format(data_reg.shape,data_clf.shape))
data_s_v = pd.read_excel('./var_range.xlsx')
#编码长度
dna_size = int(np.ceil(np.log2(np.max((data_s_v['max']-data_s_v['min'])/data_s_v['Δ值'])))) ##
值为 8
print('dna_size:',dna_size)
print('length of data_reg:',len(data_reg))
## 回归用的 20 个变量数据在分类用数据中的索引
vector_v_index = [141, 244, 314, 33, 242, 15, 17, 199, 19, 354, 357, 236, 168, 346, 294, 243,
```

```
353, 29, 35, 226]
# 利用遗传算法获得优化的分子描述符的取值范围
res = np.zeros((len(data_reg),len(data_reg.columns)+2)) #创建用来存储已优化的的特征数据，一个是 ADMET，另一个是 pIC50，用来反应算法的优化效果
res_clf = np.zeros((len(data_clf),len(data_clf.columns))) #创建用来存储已优化的的特征数据，一个是 ADMET，另一个是 pIC50，用来反应算法的优化效果
t1 = time.time()
for i in range(len(data_reg)):
    print('训练进度:',i)
    GG = GeneticGroup(np.array(data_reg.iloc[i,:]),np.array(data_clf.iloc[i,:]),vector_v_index,dna_size,data_s_v,

n_generations=500,model_r=RF_reg,model_c1=RF_clf_Caco_2,model_c2=RF_clf_CYP3A4,model_c3=RF_clf_hERG,
                      model_c4=RF_clf_HOB,model_c5=RF_clf_MN)##创建一个实例化的对象
    res[i,:-2],res_clf[i,:],zy=GG.optimization()
    res[i,-2]=RF_reg.predict(res[i,:-2].reshape(1,-1)) ##保存模型预测的回归的具体值，方便与原始数据进行对比
    res[i,-1]=RF_clf_Caco_2.predict(res_clf[i,:].reshape(1,-1)) + RF_clf_CYP3A4.predict(res_clf[i,:].reshape(1,-1)) + (RF_clf_hERG.predict(res_clf[i,:].reshape(1,-1))^1)+RF_clf_HOB.predict(res_clf[i,:].reshape(1,-1))+(RF_clf_MN.predict(res_clf[i,:].reshape(1,-1))^1)
t2 = time.time()
print('优化过程消耗的时间：',t2-t1)
pd_res = pd.DataFrame(res)
pd_res.columns = list(data_reg.columns)+['pIC50','ADMET']
# pd_res.to_excel('./优化结果_100t3.xlsx',index=False) ##t1 耗费时间 3090s,t2 耗时 2208s，###保存第一个样本的迭代过程
pd_zy = pd.DataFrame(zy)
pd_zy.columns = list(data_reg.columns)
# pd_zy.to_excel('./' + str(i) + '号样本的优化过程.xlsx',index=False)
```