2021-2022 学年第 2 学期

数学建模创新实践教育

课程论文

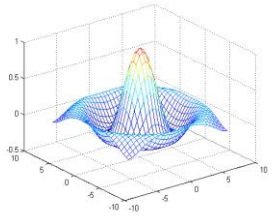
学号：10193657　04191336　06192081

姓名：李春阳　禚佳轩　胡钧耀

教师：王志俊

成绩：

五 一 数 学 建 模 竞 赛



题 目： **矿石加工质量控制问题**

摘　要：当今“双碳”理念深入人心，对于矿石加工企业来说，优化原矿的加工工艺，从而提高矿石加工合格率，就可以为节约矿产资源做出贡献。本文主要使用机器学习和智能优化相关算法建立模型，对矿石加工质量控制进行合理预测。

在问题一中，构建了**多输出支持向量机**模型，引入**投票法**对进行平均，通过搭配不同的核函数实现差异化学习，使用投票法后，最终的预测结果为：当系统温度分别为 1404.89℃，859.77℃时，四项指标分别为 **80.12，23.12，11.35，16.68**；当系统温度分别为 1151.75℃，859.77℃时得到的指标分别为 **79.81，23.39，11.67，15.88**。

在问题二中，构建**集成学习**模型，引入 **XGBoost** 和 **CatBoost 算法**，平衡两个算法之后，进一步完善模型得到结果：当产品的四项指标分别为 79.17，22.72，10.51，17.05 时，对应的两个系统温度分别为 **1315.82**℃，**803.08**℃；当产品的四项指标分别为 80.10，23.34，11.03，13.29 时，对应的两个系统温度分别为 **1346.93**℃，**803.08**℃。

在问题三中，利用**对抗学习**思想，合理利用预测得到的假数据误差，构建在误差允许范围内的随机森林分类模型，利用改进的预测模型和对抗分类模型，给出合格率预测结果：当系统温度分别为 341.40℃，665.04℃时，产品合格率为 **25.86**%；当系统温度分别为 1010.32℃，874.47℃时，产品合格率为 **12.47**%。

在问题四中，使用**粒子群**优化搜索算法，弥补上述模型寻找逆映射的困难。如果要达到 80%的产品合格率，系统温度应分别为 **1394.73**℃，**906.32**℃时；但如果要达到 99%的产品合格率，在当前现有设定的温度系统中几乎**不可实现**。为了检验模型，绘制了 **ROC 曲线**，其面积达到 0.91，模型效果较好。

本文设计多种机器学习算法，并适当进行组合，采用集成学习策略降低了学习过拟合的可能性，使结果更加准确。

关键词：支持向量机，投票法，XGBoost，CatBoost，对抗学习，粒子群

一、问题背景与重述

1.1 问题背景

在绿色低碳发展的背景下，碳达峰与碳中和的“双碳”的理念越来越受到重视，当前我国已进入新的发展阶段，“双碳”目标紧迫且艰巨，这需要全体社会各界，特别是与环境相关的企业承担起这份责任。对于矿石加工企业来说，优化原矿的加工工艺，从而提高矿石加工合格率，就可以直接或间接地节约不可再生的矿物资源以及加工所需的能源，从而为节能减排助力。矿石加工是一个复杂的过程，在加工过程中，电压、水压、温度作为影响矿石加工的重要因素，直接影响着矿石产品的质量。矿工加工实际流程如下图所示。

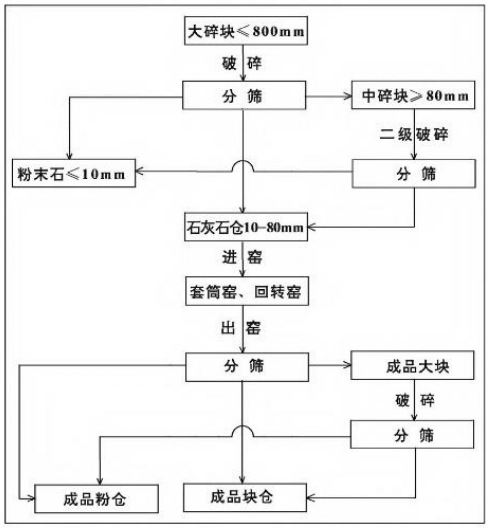


图 1.1-1 矿工加工实际流程

为了方便建模，假设矿石加工过程需要经过系统 I 和系统 II 两个环节，两个环节不分先后，其他条件（电压、水压等）保持不变。简化工艺流程如下图所示。

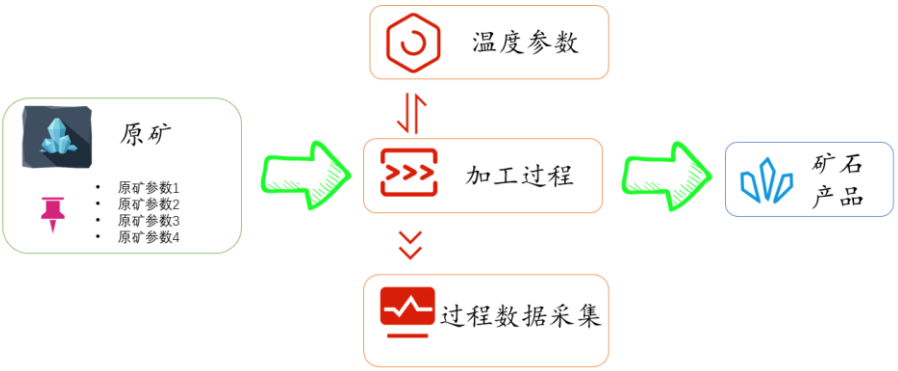


图 1.1-2 矿工加工实际流程

1.2 问题重述

针对该生产车间 2022 年 1 月 13 日至 2022 年 4 月 7 日的生产加工数据，进行数学建模，完成下列问题：

(1) 问题一要求对附件 1 中生产加工数据，研究原矿参数和系统设定温度对产品质量指标之间的关系，建立数学模型，给出利用系统温度预测产品质量的方法，并预测对应温度的产品质量指标。

(2) 问题二要求进一步探究原矿参数，系统设定温度，产品质量指标之间的关系，在问题一的基础上，分析问题一的逆映射，探究产品目标质量所对应的系统温度并预测对应产品质量指标下可能的系统温度。

(3) 问题三要求利用附件 2 的数据，通过增加过程数据，利用过程数据改进问题一模型，给出指定系统设定温度，预测矿石产品合格率的方法，并给出合格率预测结果。

(4) 问题四要求在问题三基础上，进一步分析灵敏度和准确性。判断能否达到 2022 年 4 月 10 日和 2022 年 4 月 11 日产品的合格率要求，如果可以达到，给出系统设定温度。

二、问题分析

2.1 题目整体分析

首先将附件所给的数据进行数据清洗，进行整合，整理出系统温度、产品指标以及原矿参数对应小时的数据样本。处理过程中，我们对一小时内的系统温度取均值作为对应时间的数值，而原矿参数设置为当天内 24 小时原矿参数相同。

对于该问题，我们首先需要观察数据的分布情况，探究数据之间的相关性，挖掘其数据本身的特征。为此我们选取四项指标的检测数据进行绘图。四项指标的趋势图如下图所示，从图中可以看出，参数 A，B，C 的波动范围小，D 波动范围大，指标对外界因素的变换更加敏感，可能更容易受到影响。

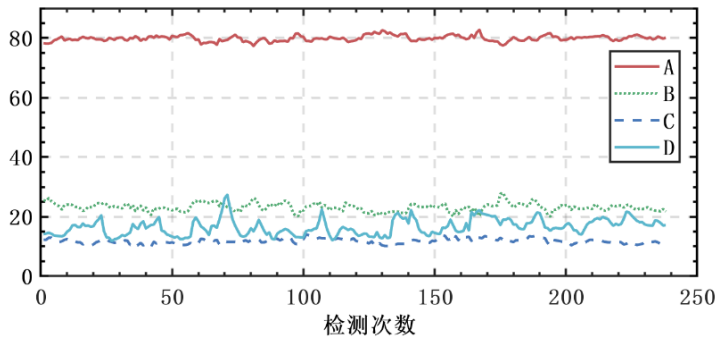


图 2.1-1 四种矿石产品检测指标时序变化图

选取两个系统温度为变量按时序进行绘图。两个系统温度的趋势图如下图所示，可以看出系统温度波动幅度都较大，特别是温度 1，波动范围较广，可能是主要影响矿石产品质量的因素之一。

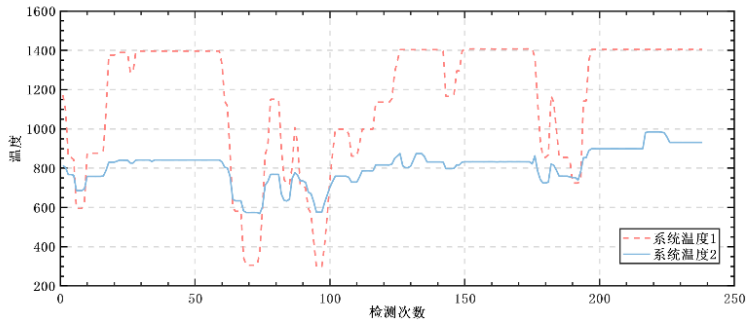


图 2.1-2 两个系统温度的时序变化图

选取四个原矿参数为变量按时序进行绘图。四项原矿参数的趋势图如下图所示，由图可以看出，原矿参数 1 变化幅度较大，原矿参数 2，3 呈现负相关，原矿参数 4 较为平缓。因此，后面重点探究原矿参数 1，2，3 对其结果的影响。

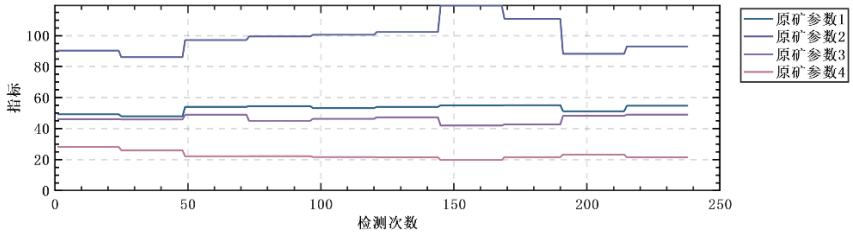


图 2.1-3 四项原矿参数的时序变化图

为了进一步探究，绘制两两变量之间散点图和核密度估计图。如下图所示，右上角为两两变量之间的散点图，左下角为核密度估计，对角线为变量本身特征。可以发现其变量之间存在多维度影响，难以直接用线性关系进行拟合。变量之间相关性不是很明显，其中指标 A，B 负相关，系统温度 1 与系统温度 2 相关性明显，原矿参数之间互相影响。

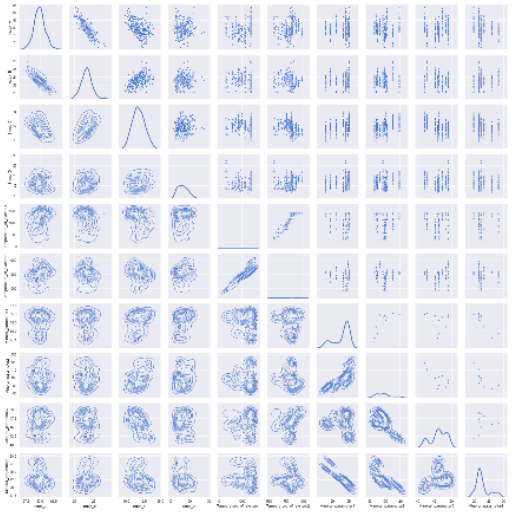


图 2.1-4 两两变量之间散点图和核密度估计图

2.2 问题一的分析

附件 1 给出了以往时间的原矿加工系统温度数据，问题 1 要求在给定原矿参数和系统温度下，给出产品质量预测结果。根据对数据样本的分析，针对样本高维特征和样本数量不多的特点，在问题一中，决定采用支持向量机进行回归预测，可以有效挖掘高维度特征之间的关系，针对问题一的产品质量指标 A，B，C，D 构建多输出支持向量机模型（MSVR）。由于样本数据偏小的原因，为了增强鲁棒性，引入投票法进行平均（Voting），通过搭配不同核函数实现差异化学习，使用投票法后，最终的预测结果是多个回归模型预测结果的平均值。

2.3 问题二的分析

利用附件 1 数据，假设原矿参数和产品目标质量已知，需要估计产品目标质量所对应的系统温度。通过问题一的分析，我们了解到矿石加工过程中，不同变量和产品质量指标之间的关系，因此在问题二中，可以探究其逆映射，但是针对支持向量机非线性问题的核函数的选择没有通用标准，难以选择一个合适的核函数，同时，问题一多项式核 poly，高斯核函数 rbf 对非线性内核存在误差，因此，引入随机森林发展而来的集成学习（Ensemble Learning），在问题一模型上进行改进。通过对比各类集成学习，最后确定引入 XGBoost 和 CatBoost 两个算法，用以平衡算两个法的性能。进一步完善模型，根据模型可以很轻易求出问题一、二的结果。

2.4 问题三的分析

附件 2 给出了生产车间一段时间内的生产加工数据及过程数据，满足下表销售条件的产品视为合格产品。问题需要在给定系统设定温度和原矿参数、过程数据的情况下，预测矿石产品合格率。

表 2.4-1 矿石合格标准

指标	指标 A	指标 B	指标 C	指标 D
销售条件	77.78 - 80.33	<24.15	<17.15	<15.62

对附件 2 的数据做同样的数据分析，拥有相同的样本特征，利用给出合格标准，统计样本合格率，结果如下图所示。由观察可知，矿石产品的整体合格率偏低，如果直接利用模型进行预测的产品质量指标按标准进行归类合格，其预测存在一定误差，在误差范围内容易影响结果。设计对抗学习，利用问题二改进的模型进行回归预测，生成产品质量指标，与原产品质量指标进行混合，构建分类模型。利用对抗思想，评估生成的样本数据与真实数据的接近程度作为评估标准，通过对比，合理利用误差，得到在误差允许范围内的分类模型（这里选用随机森林），利用改进的预测模型和对抗分类模型，给出合格率预测结果。

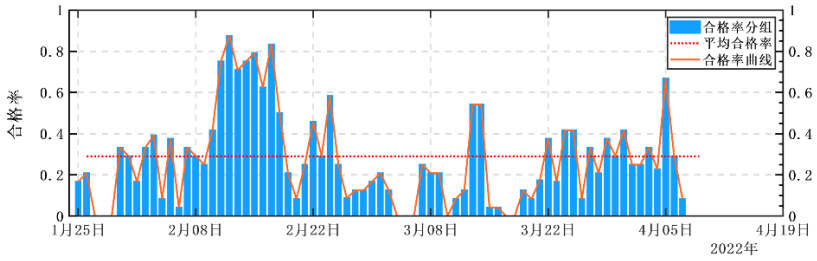


图 2.4-1 产品样本合格率时序图

2.5 问题四的分析

利用附件 2 的数据，给出在指定合格率的条件下，设定系统温度的方法，并对模型的准确性和敏感性进行分析。

由于整体合格率偏低，在 80%以上样本数较少，在前述问题中建立的模型基础上求其逆映射存在困难，这里采取优化算法来寻找结果。构建非线性映射函数，将合格率作为因变量进行研究。因为自变量温度的取值有无限种取值可能，所以对于此全局寻优问题，我们采用启发式算法中的粒子群优化算法（PSO）求解。如果无解，则代表达不到题目所给定合格率要求。

三、模型假设及符号说明

3.1 模型假设

我们已经结合实际情况对本问题进行了系统且具体的分析，根据上述分析我们做出如下合理的假设，以期利用恰当的数学模型对该问题进行详细的解答。

- 假设系统温度与调温指令设定的温度相同；
- 假设每次温度调节后的两个小时不会输入新的调温指令，系统温度变化幅度不大，可以用两小时的平均值来表示这两小时内任一时刻的温度；

3.2 符号说明

表 3.2-1 论文使用的主要符号

符号	含义
$E(X)$	数据X的期望
σ_X	数据X的方差
$Cov(X,Y)$	数据X，Y的协方差
ρ_{XY}	数据X，Y的相关性
$P(M)$	事件M发生的概率
T	系统温度
Mp	原矿参数
Pp	过程数据
C	惩罚因子
ξ_i	松弛因子
$K(x_i, x_j)$	核函数
MAE	平均绝对误差

符号	含义
MSE	均方误差
$MAPE$	平均绝对百分误差
R^2_Score	拟合度指数
$H(X_m)$	基尼不纯度
$\boldsymbol{x}_i$	第i个粒子在D维空间的位置
$\boldsymbol{v}_i$	第i个粒子在D维空间的速度
$\boldsymbol{p}_i$	第i个粒子自身搜索到的最优解
$\boldsymbol{p}_g$	整个群体搜索到的最优解

四、模型的建立与求解

4.1 问题一模型的建立与求解

4.1.1 问题一模型的预备

在矿石加工过程中，影响矿石加工质量的因素有很多，如电压、水压、温度、原矿参数等。分析题意与附件中的数据可知，其他条件（电压、水压等）保持不变，因此我们可以忽略掉其他不确定参数的影响，只考虑原矿参数的 4 个指标和系统 I、II 分别设定的温度对产品质量的影响。矿石加工过程需要经过系统 I 和系统 II 两个环节，两个环节不分先后，因此我们可以认为同一时间段内，系统 I 设定温度与系统 II 设定温度对另一系统的影响忽略不计。

对系统 I、II 分别设定的温度，原矿参数以及产品质量的 4 个指标进行相关性分析得下图。

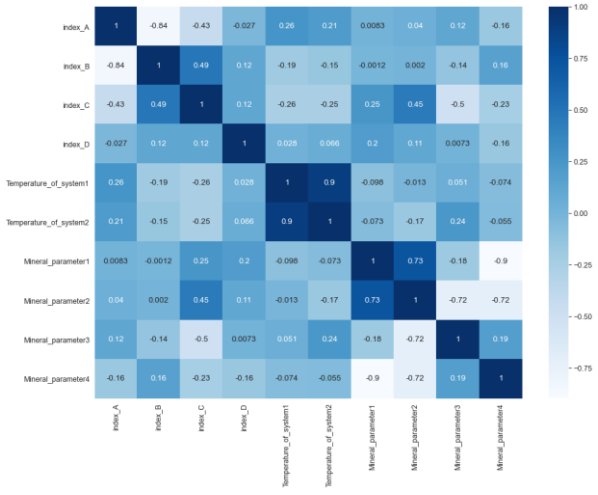


图 4.1-1 影响矿石加工因子的相关性分析

我们需要根据原矿参数的 4 个指标和系统 I、II 分别设定的温度共 6 个参数预测产品质量的 4 个指标，属于多对多的回归问题，可以转化为多个多元回归问题解决。

对于多对多的回归问题,可以转化为多个多元回归问题来解决。但对求解各

温度与原矿参数下矿石加工产品质量问题，每个温度和原矿参数下的因变量，即产品参数的各指标之间存在着一定的联系，如果分别建立自变量温度和原矿参数与其指标因变量的回归关系式,会丢失各因变量指标之间的相关信息。基于上述，我们构建多输入多输出 M-SVM 模型。[1][2]

4.1.2 SVM 模型及其改进

传统 SVM 算法假设分类问题在 H 上是线性可分的, 那么在H空间中构造最优超平面为：

$$f(x) = [w \cdot \varphi(x)] + b = 0 \tag{1}$$

为将结构风险降低到最小，将求解的初始优化问题转化为：

$$min\, J = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{2}$$

$$\begin{cases} s.t.\,|y_i - \mathrm{w}\cdot\varphi(x_i) - \mathrm{b}| \le \varepsilon + \xi_i \\ |-\mathrm{w}\cdot\varphi(x_i) + \mathrm{b} - y_i| \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, \mathrm{i} = 1, \cdots, \mathrm{n} \end{cases} \tag{3}$$

其中，$\|w\|$表示函数的复杂程度；ε为不敏感损失函数参数；ξ_i, ξ_i^*表示松弛因子,C 为惩罚因子，表示对错分样本的惩罚程度, C值越大, 对目标函数的损失也越大。上式的求解在保持基于VC维的上界小的基础上，通过最小化 $\sum_{i=1}^{n}\xi_i$ 达到经验风险最小化。引入松弛变量能够消除个别样本点对分类器的不良影响, 在训练错误和泛化能力间有所折中, 所以它具有一定的鲁棒性。满足式中约束条件的最小松驰变量ξ_i为：

$$\xi_i = \max\left(0, 1 - y_i(w \cdot \varphi(x_i) + b)\right) \tag{4}$$

(1) 当$\xi_i = 0$时，表示样本点位于分类边界之外或者在边界上，分类正确；

(2) 当$0 < \xi_i \le 1$时，表示样本点位于分类面内, 分类正确；

(3) 当$\xi_i > 1$ 时，样本点被错误分类。

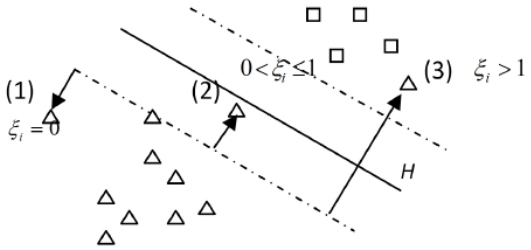


图 4.1-2 松弛变量

将式(1)优化问题转变为凸二次优化问题，引入拉格朗日乘数法，

$$L(\omega, \boldsymbol{b}, \xi, \alpha, \alpha^*, \gamma, \gamma^*) = \frac{1}{2}||\omega|| + C\sum_{i=1}^{n}(\xi_i^* + \xi) - \sum_{i=1}^{n}\alpha_i\left(\xi_i + \varepsilon - |\boldsymbol{y_i}| + f(x_i)\right) - \sum_{i=1}^{n}\alpha_i^*\left(\xi_i^* + \varepsilon - |\boldsymbol{y_i}| + f(x_i)\right) - \sum_{i=1}^{n}(\xi_i\gamma_i - \xi_i^*\gamma^*) \tag{5}$$

将式(1)转换为对偶形式来提高收敛速度，转换如下：

$$W(\alpha,\ \alpha^*) = -\frac{1}{2}\sum_{i,=1}^{n}(\alpha_i - \alpha_i^*)\left(\alpha_j - \alpha_j^*\right)\left(\varphi(x_i)\ \varphi\left(x_j\right)\right) + \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)|y_i| - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\ ; \tag{6}$$

$$\begin{cases} s.t.\left|\left|w\right|\right| = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\,x_i \\ \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \\ 0 \le \alpha_i, \alpha_i^* \le C \end{cases} \tag{7}$$

对于线性回归问题，SVM 回归函数为

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\left(\varphi(x_i), \varphi(x)\right) + b \tag{8}$$

为防止维数灾难发生，用核函数$k(x_i, \boldsymbol{x})$代替高维空间中的向量内积$\left(\varphi(x_i), \varphi(\boldsymbol{x})\right)$，则 SVM 的回归函数为：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)k(x_i, \boldsymbol{x}) + \boldsymbol{b} \tag{9}$$

根据 Hilbert-Schmidt 理论，核函数$K\left(x_i, x_j\right)$是满足 Mercer 条件的任意对称函数。由于样本数据偏小的原因，为了增强鲁棒性，引入投票法进行平均，通过搭配不同核函数实现差异化学习。常见的核函数如下：

(1) poly 多项式核函数：

$$K\left(x_i, x_j\right) = \left(x_i^T x_j + 1\right)^d \tag{10}$$

其中，d 表示多项式的梯度。

(2) Sigmoid 感知核函数：

$$K\left(x_i, x_j\right) = tan\ h\left(w x_i^T x_j + r\right) \tag{11}$$

(3)高斯核函数：

$$K\left(x_i, x_j\right) = \exp\left(-\frac{\left(x_i - x_j\right)^2}{2\sigma^2}\right) \tag{12}$$

由于支持向量机允许使用不同的核函数，即允许使用不同的假设空间，所以它在解决多样应用问题时，具有一定的柔韧性。显然，支持向量机的鲁棒性和柔韧性也是我们设计解决、问题算法时所渴求的。然而传统的 SVM 只能解决二值问题，针对多输入输出目标预测问题，还需对其进行改进。

✧ **改进的多输入多输出 M-SVM 模型**

针对多类问题，弥补现有多类支持向量机算法的不足，我们采用多类支持向量机 M-SVM 方法。[3][4]首先将所有矿石加工参数数据集分成两个子类,再将子类进一步划分成两个次级子类，如此循环直到每个子类只包含一个单独的类别为止，包含了不同类别的子类作为层次树的分支结点，只包含一类样本的子类作为层次

树的叶子结点，从而形成了层次树结构模型。从某种程度上说，层次树模型是一种先验知识，其作用是指导支持向量机对待测样本做最后的分类。多类支持向量机的训练过程如下图所示。对于待测样本，先从根结点分类器对其进行划分，根据判别函数将其归为左子结点或者右子结点，逐层往下直至待测样本x被分配到某个叶子结点，则将待测样本x归到叶子结点所属的类别，分类过程结束.

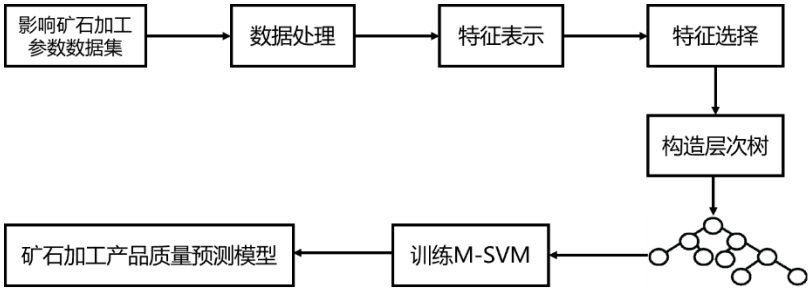


图 4.1-3 M-SVM 矿石加工产品质量预测模型流程图

4.1.3 模型的求解与检验

✧ **M-SVM 模型的求解**

首先，由于支持向量机算法对非标准化数据的容忍度较差，因此我们对数据进标准化处理。我们创建数据集，进行有监督分类学习，再创建 SVR 回归器，设定参数后再创建多输入多输出 SVR 回归器，训练回归器后为测试数据生成预测，引入投票法进行平均，通过搭配不同核函数实现差异化学习，投票法最终的预测结果是多个回归模型预测结果的平均值，评估回归器，得到问题二的预测结果如下：

表 4.1-1 问题 1 结果

时间	系统 I 设定温度	系统 II 设定温度	指标 A	指标 B	指标 C	指标 D
2022-01-23	1404.89	859.77	80.12	23.12	11.35	16.68
2022-01-23	1151.75	859.77	79.81	23.39	11.67	15.88

✧ **M-SVM 模型的检验**

我们选取 MSE（均方误差）、MAE（平均绝对误差）、可解释的方差分数、R^2、MAPE（绝对平均百分比偏差）作为评估预测性能的检验标准。

MSE：如果 $\hat{y}_i$是第i个样本的预测值，y_i是与之相一致的真实值，均方误差 MAE 对n_{samples}的估计值被定义如下：

$$MSE(y,\hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2 \tag{13}$$

MAE：如果$\hat{y}_i$是第i个样本的预测值，且y_i是与之相一致的真实值，n_{samples}样本的平均绝对误差（MAE）定义如下：

$$MAE(y,\hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i| \tag{14}$$

可解释的方差分数：如果$\hat{y}_i$是标签估计值，y 是与之相一致（正确）的标签

输出，并且 Var 是 Variance，标准差的平方，则可解释的方法计算如下：

$$\text{explained }_{v}ariance(y,\hat{y}) = 1 - \frac{Var\{y-\hat{y}\}}{Var\{y\}} \tag{15}$$

最好的可能取值是 1.0，值越低越不好。

$\boldsymbol{R^2}$**:**代表方差（y 的）的比例，被解释为模型中的独立变量。它是模型的拟合度指数，因此，可以代表模型对未知数据的预测好坏程度，通过计算可解释方差的比例。如果$\hat{y}_i$是样本i的预测值, 那么y_i是与所有n个样本相一致的真实值，则R^2的估计表达式为:

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2} \tag{16}$$

MAPE：用百分比表示，与比例无关，可用于比较不同比例的预测，易于向利益相关者解释。

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|x_i-y_i|}{x_i} \tag{17}$$

其中计算后得到的结果如下所示:

表 4.1-2 检验结果

	MSE	MAE	可解释的方差分数	R^2得分	MAPE
预测 1	38634.40	146.97	0.68	0.67	0.206
预测 2	1797.27	31.57	0.83	0.83	0.043

4.2 问题二模型的建立与求解

4.2.1 模型的建立

在问题一中，我们需要根据原矿参数和系统温度求解产品质量，与问题一相比较，问题二的初始条件为原矿参数和产品目标质量已知，而系统温度未知，以此来估计产品目标质量所对应的系统温度，因此问题二的求解目标是温度，我们可以认为问题二是问题一的逆映射，对于问题二的模型，为了解决问题一多项式核 poly，高斯核函数 rbf 对非线性内核存在误差的问题，我们在问题一模型上加以改进，引入随机森林进行集成化学习。为更加精确地求解模型的结果，引入 XGBoost 和 CatBoost 平衡算法的性能。[9]

✧ **决策树与随机森林**

决策树是一种用于分类和回归的非参数有监督学习方法。其目标是创建一个模型，通过学习从数据特性中推断出的简单决策规则来预测目标变量的值。决策树具有易于理解和解释、能够处理多输出问题、对模型的可靠性作出解释等优点。`sklearn.ensemble` 集成模块包括两种基于随机决策树的平均算法：RandomForest 算法和 Extra-Trees 算法。这两种算法都是专门为树设计的扰动和组合技术。因此在分类器构造过程中引入随机性来创建一组不同的分类器的集合，集成之后的预测是每个分类器的平均。在本文中我们采用随机森林算法。随机森

林（RF）是一种基于分类树的统计学习理论，其利用 Bootstrap 重抽样方法将多个样本集从原始样本集中有放回地抽取出来，并对每个样本集分别进行决策树建模，每棵决策树随机选择特征对内部节点进行属性分裂，最终构成一片随机森林，预测结果综合每棵决策树的结果投票得出。[5]

与其他分类器一样，随机森林分类器必须拟合两个数组：一个稀疏或密集的 X 数组，包含训练样本；数组 Y，包含训练样本的目标值（类标签）。和决策树一样，森林也扩展到多输出问题的数组。

在随机森林中，集成模型中的每棵树构建时的样本都是由训练集经过有放回抽样而来。随机森林是一种元估计器，它在数据集的不同子样本上匹配许多决策树分类器，并使用平均来提高预测精度和控制过拟合。

✧ **模型的构建**

给定训练向量$x_i \in R^n, i = 1, \ldots I$和标签向量 $y \in R^l$，决策树递归地划分空间，使得具有相同标签的样本被分到一样的组。让节点 m 处的数据集用 Q 表示，对于一个由特征j 和阈值 t_m 组成的候选划分数据集$\theta = (j, t_m)$，将数据划分为$Q_{left}(\theta)$和$Q_{\text{right}}(\theta)$两个子集。

$$Q_{\text{left}}(\theta) = (x, y) \mid x_j \leq t_m \tag{18}$$

$$Q_{\text{right}}(\theta) = QQ_{left}(\theta) \tag{19}$$

节点m处的不存度用不存度函数 H 计算，其选择取决于正在解决的任务(分类或回归)。

$$G(Q, \theta) = \frac{n_{\text{left}}}{N_m} H\left(Q_{\text{left}}(\theta)\right) + \frac{n_{\text{right}}}{N_m} H\left(Q_{\text{right}}(\theta)\right) \tag{20}$$

选择使不存度最小化的参数：

$$\theta^* = amearg min_{\theta} G(Q, \theta) \tag{21}$$

对子集$Q_{left}(\theta^*)$和$Q_{right}(\theta^*)$进行递归，直到达到最大允许的深度，$N_m < min_{samples}$或$N_m = 1$。

✧ **分类标准**

如果目标是变量的值 $0,1, \ldots k-1$ 的分类结果，对于节点 m，表示具有N_m个观测值的区域R_m,令:

$$p_{mk} = 1/N_m \sum_{x_i \in R_m} I(y_i = k) \tag{22}$$

p_{mk}表示的是节点 m 中 k 类观测的比例。

常见的不存度度量的方法是基尼不纯度（Gini）：

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk}) \tag{23}$$

熵(Entropy)的表达式如下:

$$H(X_m) = -\sum_k p_{mk} \log(p_{mk}) \tag{24}$$

错误分类(Misclassification)表达式如下:

$$H(X_m) = 1 - \max(p_{mk}) \tag{25}$$

其中X_m是节点 m 中的训练数据。

✧ **XGBoost**

XGBoost 是一个优化的分布式梯度增强库，旨在实现高效，灵活和便携，是在 Gradient Boosting 框架下实现机器学习算法，可以快速准确地解决许多数据科学问题。[6]

1. 梯度下降

在 GBDT 中，我们每次生成下一个弱学习器，都是把损失函数的梯度作为学习目标，相当于利用梯度下降法进行优化来逼近损失函数的最小值，也就是使得损失函数为 0，最终学习器尽可能接近真实结果。

$$F_n(x) = \sum_{i=1}^{n} f_i(x) \tag{26}$$

$$F_n(x) = F_{n-1}(x) + f_n(x) = F_{n-1}(x) - \nabla L\big(F_{n-1}(x)\big) \tag{27}$$

而在 XGBoost 中，我们则是把损失函数的二阶泰勒展开的差值作为学习目标，相当于利用牛顿法进行优化，来逼近损失函数的最小值，使得损失函数为 0 。

$$F_n(x) = \sum_{i=1}^{n} f_i(x) \tag{28}$$

$$F_n(x) = F_{n-1}(x) + f_n(x) = F_{n-1}(x) - \frac{L^{'}\left(F_{n-1}(x)\right)}{L^{''}\left(F_{n-1}(x)\right)} \tag{29}$$

2. 正则项

正则项是为了防止模型过拟合。于是，一般的损失函数 L(x)就变成了目标函数$L(x) + \Omega(x)$, 随着树的复杂度增大，对应的目标函数也就变大，这样就有效防止了过拟合。叶子节点个数(T)，叶节点分数(w)

$$\Omega = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{30}$$

对叶子节点个数进行惩罚，相当于在训练过程中做了剪枝。

将 XGboost 的目标函数进行化简，并把 $f(x) = w_q(x)$ 决策树和 $\Omega = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$ 代入:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \widehat{y_i^{(t-1)}} + f_t(\mathbf{x_i})\right) + \Omega(f_t) \tag{31}$$

$$\begin{aligned}\widetilde{\mathcal{L}^{(t)}} &= \sum_{i=1}^{n}\left[g_i f_t(\mathbf{x_i}) + \frac{1}{2}h_i f_t^2(\mathbf{x_i})\right] + \Omega(f_t) \\ &= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right)w_j^2\right] + \gamma T \\ &= \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}\left(H_j + \lambda\right)w_j^2\right] + \gamma T\end{aligned} \tag{32}$$

令其导数为 0 ，解得每个叶节点的最优预测分数为:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{33}$$

代入目标函数，得到最小损失为:

$$\widetilde{L^*} = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \tag{34}$$

✧ **CatBoost**

CatBoost 是一种基于对称决策树算法的参数少、支持类别型变量和高准确性的 GBDT 框架，可以有效提升提高算法的准确性和泛化能力。[7]

假设需要向融合添加新树。需要评分函数才能在候选树之间进行选择。给定一个候选树 f，让a_i表示$f(x_i)$，w_i表示对象的权重，g_i表示的相应梯度。考虑以下分数函数：

$$L2 = -\sum_i w_i \cdot (a_i - g_i)^2 \tag{35}$$

$$Cosine = \frac{\sum w_i \cdot a_i . g_i}{\sqrt{\sum w_i\ a_i^2}\ \cdot \sqrt{\sum w_i\ g_i^2}} \tag{36}$$

假设需要找到深度为 1 的树 f 的结构。这种树的结构由某些特征的索引 j 和边界值 c 决定。设 $x_{i,\ j}$ 是第 i 个对象上第 j 个特征的值，a_{left}和 a_{right}是 f 叶子处的值。如果$x_{i,\ j} \le c$，$f(x_i) = a_{left}$，如果$x_{i,\ j} > c$,现在的目标是根据所选的分数函数找到最佳的 j 和 c.对于 L2 分数函数，公式采用以下形式：

$$\begin{aligned} S(a,g) &= -\sum_i w_i(a_i - g_i)^2 \\ &= -\left(\sum_{i:x_{i,j}\le c} w_i\left(a_{left} - g_i\right)^2 + \sum_{i:x_{i,j}>c} w_i\left(a_{right} - g_i\right)^2\right) \end{aligned} \tag{37}$$

$S(a,g)$表示$W_{left} = \sum_{i:\ x_{I,\ j}\le c} w_i$和 $W_{right} = \sum_{i:\ x_i > c} w_i$的最佳加权平均值：

$$a_{left}^* = \frac{\sum_{i:x_{i,j}\le c} w_i g_i}{W_{left}} \tag{38}$$

$$a_{right}^* = \frac{\sum_{i:x_{i,j}>c} w_i g_i}{W_{right}} \tag{39}$$

在扩展括号并删除术语后，这些术语在优化中是恒定的：

$$j^*, c^* = argmax_{j,c} W_{left} \cdot \left(a_{left}^*\right)^2 + W_{right} \cdot \left(a_{right}^*\right)^2 \tag{40}$$

后者的 argmax 可以通过蛮力搜索来计算。

CatBoost 提供了以下方法来影响得分：模型中首次出现的特征的每个特征的惩罚。如果当前候选项是第一个在模型中包含特征的候选项，则从分数中减去给定值。首次使用对象功能时对每个对象的惩罚。给定值乘以除以当前拆分并首次

使用该功能的对象数。最终得分的计算方法如下：

$$Score' = Score \cdot \prod_{f \in S} W_f - \sum_{f \in S} P_f \cdot U(f) - \sum_{f \in S} \sum_{x \in L} EP_f \cdot U(f,x) \tag{41}$$

其中，W_f是特征权重，P_f是每个功能的惩罚，EP_f是每个对象的惩罚，S 是电流分割，L 是当前叶子

$$U(f) = \begin{cases} 0, \text{如果} f \text{已在模型中使用} \\ 1, \text{其他} \end{cases} \tag{42}$$

$$U(f,x) = \begin{cases} 0, \text{如果} f \text{已在用于对象} x \\ 1, \text{其他} \end{cases} \tag{43}$$

4.2.2 模型的求解与检验

在求解问题二时我们沿用问题一中的 MSVM 模型，先对数据进标准化处理，再创建数据集，进行有监督分类学习，再创建 SVR 回归器，设定参数后再创建多输入多输出 SVR 回归器与多输入输出随机森林回归器，训练回归器后为测试数据生成预测数据集，利用 XGBoost 与 CatBoost 平衡算法性能，引入投票法进行平均，通过搭配不同核函数实现差异化学习，投票法最终的预测结果是多个回归模型预测结果的平均值，评估回归器[10]，得到问题二的预测结果如下表所示：

表 4.2-1 问题 2 结果

时间	指标 A	指标 B	指标 C	指标 D	系统 I 设定温度	系统 II 设定温度
2022-01-24	79.17	22.72	10.51	17.05	1351.82	803.08
2022-01-24	80.10	23.34	11.03	13.29	1346.93	803.08

我们继续沿用问题一的检验标准，选取 MSE（均方误差）、MAE（平均绝对误差）、可解释的方差分数、R^2、MAPE (绝对平均百分比偏差)作为评估预测性能的检验标准，计算后得到的结果如下所示：

表 4.2-2 检验结果

	MSE	MAE	可解释的方差分数	R^2得分	MAPE
预测 1	60867.46	173.45	0.49	0.48	0.28
预测 2	3882.98	44.36	0.65	0.64	0.06

检验结果达到我们的预期标准，因此认为模型的预测结果可靠性较高。

4.3 问题三模型的建立与求解

4.3.1 模型的建立

✧ **对抗思想**

生成式对抗网络模型(Generate Adversial Network, GAN)最直接的用途是生成数据，而数据质量的好坏则是评判 GAN 成功与否的关键。基本的 GAN 网络结

构主要由两部分构成：一部分是生成模型G，主要是来生成假输入数据（Fake Data）；另一部分是判别模型D，利用假输入数据和数据集中的真实数据来训练一个分类模型。其主要思想是将判别器和生成器这两个网络的对抗和博弈，在对抗中进步，使生成数据逼近真实数据。[8]

结合 GAN 的对抗学习思想，我们将系统温度、原矿参数、过程参数作为自变量，使用问题二得到的预测模型进行回归预测，生成虚拟的产品质量指标，与原产品质量指标进行混合，构建分类模型。最后评估生成样本和真实样本的相似度，作为合格标准，得到在误差允许范围内的随机森林分类模型。

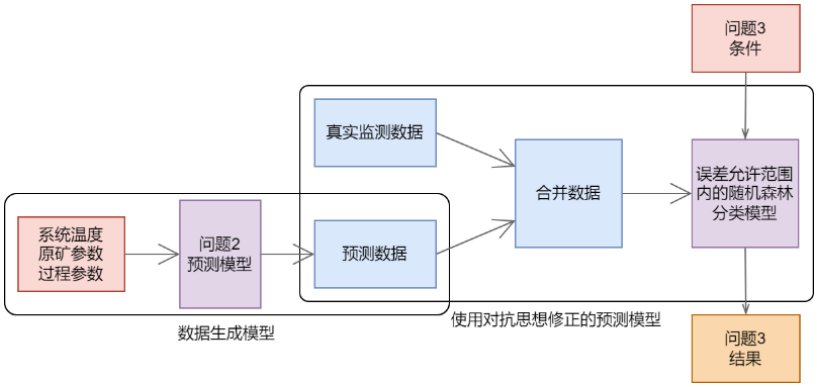


图 4.3-1 GAN 思想在本题中的使用方法

✧ **模型评估检验指标**

1. 平衡精度（balanced_accuracy_score）

在二元和多类分类问题中，平衡精度可以处理不平衡的数据集。它定义为每个类获得的召回率的平均值。

2. 混淆矩阵（confusion_matrix）

混淆矩阵是表示精度评价的一种标准格式，用$n \times n$的矩阵形式来表示。可以把分类结果的精度显示在一个混淆矩阵中。对于如下的n个类别的混淆矩阵：

$$\boldsymbol{A}_{n\times n}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{44}$$

$a_{i,j}$表示i类的样本被认为是j类的数量，$a_{i,i}$越大，则模型效果越好。

3. 平均精度（average_precision_score）

平均精度会预测值的平均准确率，该分值对应于 Precision/Recall 曲线下的面积。P_n是类别为n的精确率，R_n是类别为n的召回率。

$$\text{Precision} = \frac{TP}{TP+FP} \tag{45}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{46}$$

$$\text{AP} = \sum_n (R_n - R_{n-1}) P_n \tag{47}$$

4. 分类准确率（accuracy_score）

分类准确率分数是指所有分类正确的百分比，即找出测试值与预测值不同总数除以测试值总数。

$$\text{accuracy} = \sum_{i}^{n} y_{\text{test},i} = y_{\text{predict},i} / n \tag{48}$$

4.3.2 模型的求解

合并数据后选择随机森林 `RandomForestClassifier` 模型，决策树个数选择 `n_estimators = 10`，编程进行模型的建立与计算，预测结果如下所示。

表 4.3-1 问题 3 结果

时间	系统 I 设定温度	系统 II 设定温度	合格率
2022-04-08	341.40	665.04	0.2586
2022-04-09	1010.32	874.47	0.1247

最终得到的模型评估参数如下所示。

表 4.3-2 问题 3 模型评估参数

平衡精度	混淆矩阵	根据预测分数计算平均精度	精度分类得分
0.9179	$\begin{bmatrix} 646 & 37 \\ 34 & 275 \end{bmatrix}$	0.8187	0.9284

检验结果达到我们的预期标准，因此认为模型的预测结果可靠性较高。

4.4 问题四模型的建立与求解

4.4.1 前述模型的敏感度分析

对问题三中，增加增强生成数据的模型随迭代次数变化的产品四项指标的 R^2_Score 和平均绝对百分误差（MAPE）的变化如下图所示。观察可知：随着迭代次数的增加，增强分类模型下各项指标的 R^2_Score 不断提升，但在迭代次数为 100 次以后提升速率明显下降；产品各项指标的平均绝对百分误差不断下降，但同样在迭代次数为 100 次以后下降速率明显降低，优化效果不再明显。

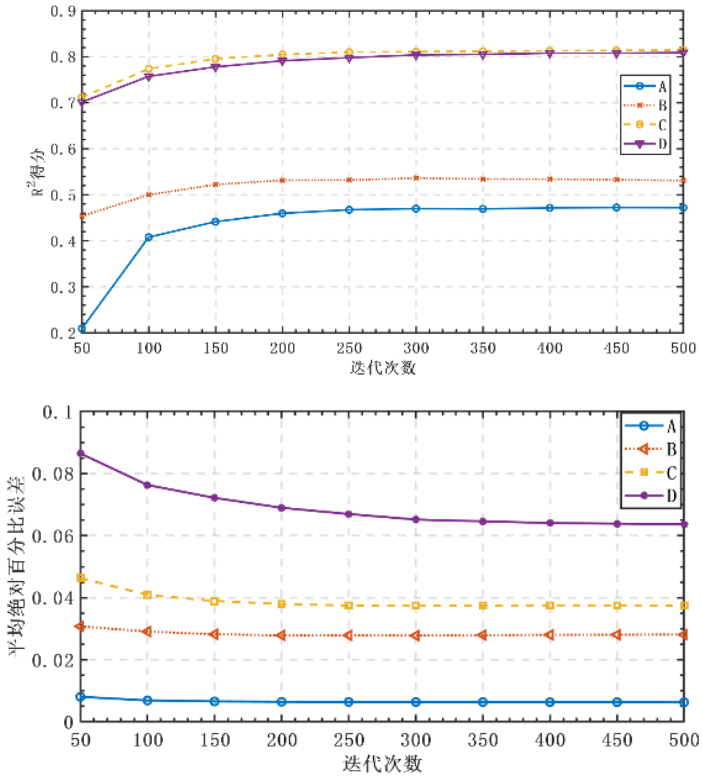


图 4.4-1 增强数据模型的 R2 分数（上）和 MAPE（下）随迭代次数的变化

对问题三中，增加增强生成数据的分类模型进行参数调整：在迭代次数为 100 时调整随机森林的深度，以及在随机森林深度为 3 时，调整模型迭代的次数。其产品四项评价指标变化如下图所示。观察可知，综合考虑时间效率与优化能力，在深度设置为 3 时，迭代次数为 100 时，精度分类得分与平均精度都较高，模型可以达到最优效果。

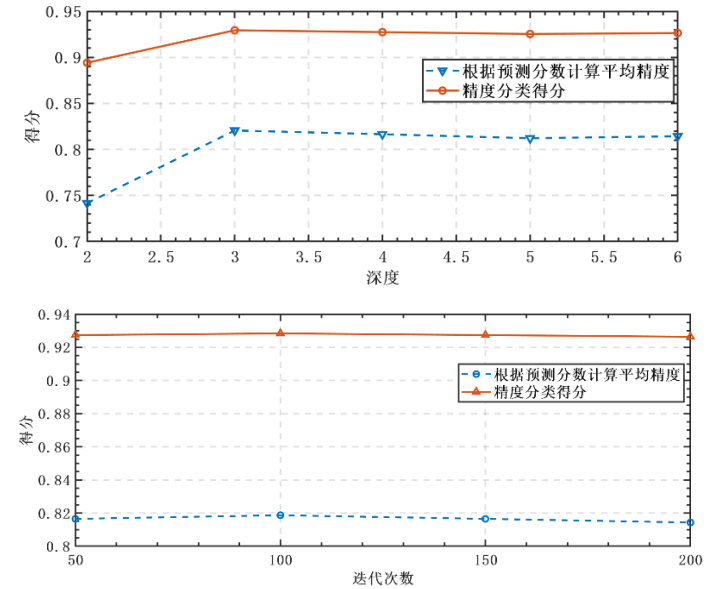


图 4.4-2 增强数据模型的平均精度与分类得分随模型深度（上）与迭代次数（下）的变化

4.4.2 模型的建立

由于整体合格率偏低，在 80%以上样本数较少，在前述问题中建立的模型基础上求其逆映射存在困难，这里采取优化算法来寻找结果。构建非线性映射函数，将合格率作为因变量进行研究。因为自变量温度的取值有无限种取值可能，所以对于此全局寻优问题，我们采用启发式算法中的粒子群优化算法（PSO）求解。如果无解，则代表达不到题目所给定合格率要求。

✧ **粒子群算法（PSO）**

粒子群优化算法是一种基于群体物种或者粒子研究的随机优化方法，其思想来源于人工生命和演化计算理论。

基本的全局 PSO 算法如下所述。在以D维度的目标搜索空间中，有M个粒子组成一个粒子集群，其中第i个粒子表示为一个D维的向量$\boldsymbol{x}_i=(x_{i1},\ x_{i2},\ ...,\ x_{iD}),\ i=1,\ 2,\ ...,\ m$，即第$i$个粒子表示为$D$维的搜索空间中的位置记为$\boldsymbol{x}_i$，每个粒子所处的坐标都可以认为是可能的解，将$\boldsymbol{x}_i$带入设计好的目标函数，即可计算出这个粒子在当前坐标的适应值，根据适应值的大小衡量$\boldsymbol{x}_i$的优劣。第i个粒子的移动速度或者步长表示为一个D维的向量$\boldsymbol{v}_i=(v_{i1},\ v_{i2},\ ...,\ v_{iD})$。对于最优位置有两种，一种是第$i$个粒子自身搜索到的最优位置，第二种是整个群体搜索到的最优位置，分别为$\boldsymbol{p}_i=(p_{i1},\ p_{i2},\ ...,\ p_{iD})$和$\boldsymbol{p}_g=(p_{g1},\ p_{g2},\ ...,\ p_{gD})$。对粒子执行如下操作：

$$\begin{aligned} v_{id} &= v_{id}+c_1r_1(p_{id}-x_{id})+c_2r_2(p_{gd}-x_{gd}) \\ x_{id} &= x_{id}+v_{id} \end{aligned} \tag{49}$$

其中$i=1,\ 2,\ ...,\ m,\ d=1,\ 2,\ ...,\ D$；学习因子$c_1$和$c_2$是非负常数；$r_1$和$r_2$是介于$[0,1]$之间的随机数；$v_{id}\in[-v_{\max},v_{\max}]$，$v_{\max}$是自行设定的常数。如果迭代达到最大次数或者粒子群搜索到的最优值达到了设定的阈值，就结束算法，其算法流程图如下所示。

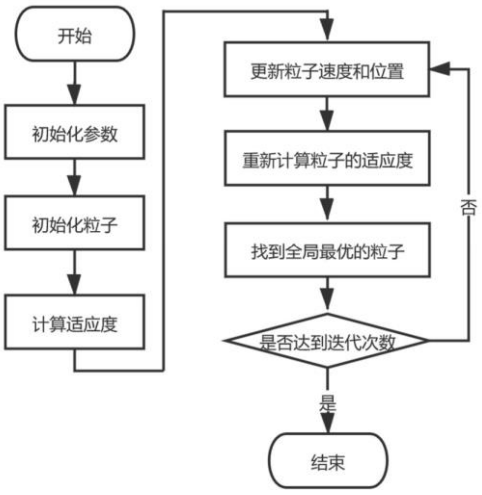


图 4.4-3 PSO 算法流程图

4.4.3 模型的求解与准确性评价

使用 PSO 算法对问题四进行求解，其结果如下所示。由于设置 99%的合格率时，PSO 算法无解，认为不达到给定要求。

表 4.4-1 问题 4 结果

时间	合格率	能否达到	系统 I 设定温度	系统 II 设定温度
2022-04-10	80%	是	1394.73	906.32
2022-04-11	99%	否	/	/

模型的准确性评价可以使用 ROC 曲线（Receiver Operating Characteristic Curve）来进行直观分析。平面的横坐标是假正率（false positive rate，FPR)，纵坐标是真正率（true positive rate，TPR)，一般情况下，这个曲线都应该处于(0, 0)和(1, 1)连线的上方，拐点的位置可以来衡量模型的优劣及其敏感度。对应地，曲线下面积（Area Under roc Curve，AUC)就是处于 ROC 曲线下方封闭图形的面积，也可其用来衡量模型的优劣。对于本模型而言，绘制 ROC 曲线如下图所示，其拐点距离（0，1）较近，且 AUC=0.91，模型效果较好。

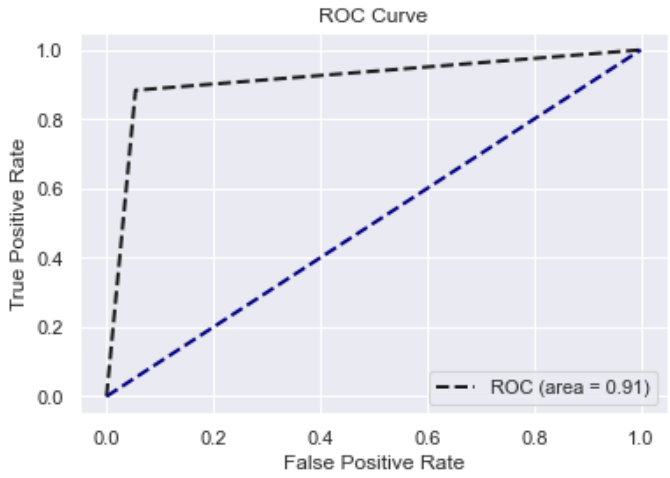


图 4.4-4 ROC 曲线

五、模型的评价与推广

5.1 模型的优点

5.1.1 问题一、二模型的优点

1. SVM：当样本量不是海量数据时，构建的模型准确率高，泛化能力强；
2. Voting：在多个机器学习算法中，进行少数服从多数原则投票，提高模型的鲁棒性和泛化能力；
3. XGBoost：的决策树生长策略是按层增长（Level-wise），它可以同时分裂同一层的叶子，支持高效并行，过拟合风险较小；
4. Catboost：可以在降低过拟合的同时，保证所有数据集都可用于学习。

5.1.2 问题三模型的优点

1. 引入对抗学习 GAN 网络的思想，以半监督方式训练分类器，使得在误差范围能更准确的预测结果；
2. 对抗学习使得数据增强，具有更广的适用性；
3. 随机森林利用构建决策树，可以处理非线性特征且考虑了变量之间的相互作用。

5.1.3 问题四模型的优点

粒子群智能优化算法方便简单，易于实现，对比其他的搜索算法，其速度更快，方便易操作。

5.2 模型的缺点

5.2.1 问题一二模型的缺点

1. 模型较为复杂，尽管通过 Voting 使得模型更加稳定，但模型也产生了大量冗余，造成模型训练速度慢，耗时长；
2. 问题一样本数据远小于问题三，使得部分模型欠拟合，只能减少测试集合，扩大训练集；
3. 由于 XGBoost 和 CatBoost 与 SVM 特性不同，在鲁棒性增强的同时，准确性有所下降。

5.2.2 问题三模型的缺点

1. 特征处理小号消耗大量内存和时间，需要在准确性和速度上进行权衡；
2. 对抗训练学习了预测出的假的数据，预测数据质量的好坏是模型成功与否的关键。

5.2.3 问题四模型的缺点

1. 有时粒子群在俯冲过程中会错失全局最优解；
2. 应用 PSO 算法处理高度复杂问题时，算法可能过早收敛；
3. PSO 算法是一种概率算法，搜索过程带有随机性。

5.3 模型的推广或改进

1. 模型二对模型一进行了改进，弥补了支持向量机的曲线，也融合了 XGBoost 和 CatBoost 的优点，在牺牲一部分准确性的情况下，使得模型更加稳定，对此可以更容易的利用在其他问题上。

2. 针对问题三中对抗学习，可以引入对抗网络和变分自编码器进行模拟，可以增强其模型稳定性和准确率，但也需要更多的样本数据，问题三中数据太少，无法构建神经网络进行预测。

3. 在问题四中我们采用了 PSO 算法，在启发式算法中，还可以升级为人工鱼群法，加入觅食行为、聚群行为、追尾行为、随机行为等控制，提高搜索算法的精确度。

六、参考文献

[1] Bao Y , Xiong T , Hu Z . Multi-Step-Ahead Time Series Prediction using Multiple-Output Support Vector Regression[J]. 2014.

[2] Li Z L . Multi-output least-squares support vector regression machines[J]. Pattern Recognition Letters, 2013.

[3] Chung W , Kim J , Lee H , et al. General Dimensional Multiple-Output Support Vector Regressions and Their Multiple Kernel Learning[J]. IEEE Transactions on Cybernetics, 2017, 45(11):2572-2584.

[4] Rai P , Kumar A , Iii H D . Simultaneously Leveraging Output and Task Structures for Multiple-Output Regression[J]. advances in neural information processing systems, 2012.

[5] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[6] Chen T , Guestrin C . XGBoost: A Scalable Tree Boosting System[J]. ACM, 2016.

[7] Hancock J T , Khoshgoftaar T M . CatBoost for big data: an interdisciplinary review[J]. Journal of Big Data, 2020, 7(1).

[8] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[J]. Advances in neural information processing systems, 2014, 27.

[9] 王露. 基于集成学习的异常数据检测及其在网络流量数据中的应用[D].华中师范大学,2021.

[10] 王雨璠. 基于两级分类模型的 Airbnb 旅行目的地分类预测[D].兰州大学,2021.DOI:10.27204/d.cnki.glzhu.2021.000362.

七、附录

本文程序用到了 MATLAB R2020a，SPSS，Python，Excel（包括一些开源的库和包，用于求解问题）本文绘图使用了 MATLAB，JavaScript，DrawIO 等软件和语言，用来美化和绘制图片。

附录 1：因变量相关性热力图绘制（MATLAB）

```
clear;clc;
load data
x=data;
for i=1:6
x(:,i)=x(:,i)/x(1,i);
end
x=x';
n=size(x,1);
ck=x(1,:);m1=size(ck,1);
bj=x(2:n,:);m2=size(bj,1);
for i=1:m1
for j=1:m2
t(j,:)=bj(j,:)-ck(i,:);
end
jc1=min(min(abs(t')));jc2=max(max(abs(t')));
rho=0.5;
ksi=(jc1+rho*jc2)./(abs(t)+rho*jc2);
rt=sum(ksi')/size(ksi,2);
r(i,:)=rt;
end
r
[rs,rind]=sort(r,'descend') %对关联度进行排序
```

附录 2：因变量相关性热力图绘制（Python）

```
sns.set(color_codes=True)
pal = sns.color_palette("viridis", 10)
sns.set_palette('muted')
data = pd.read_excel("data.xlsx")
name =
["index_A","index_B","index_C","index_D","Temperature_of_system1",
"Temperature_of_system2",  "Mineral_parameter1",
"Mineral_parameter2",  "Mineral_parameter3",
"Mineral_parameter4","Process_parameter3","Process_parameter4"]
date = data[name]
```

附录 2：因变量相关性热力图绘制（Python）

```
df = pd.DataFrame(date)
corr = df.corr()
plt.figure(figsize=(16,12))
sns.set_context('paper',font_scale=1.4)
sns.heatmap(corr, cmap='Blues', annot=True)
```

附录 3：问题一：MVSR 模型的实现（Python）

```
from sklearn.datasets import make_regression
from sklearn.multioutput import MultiOutputRegressor
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# %matplotlib inline
sns.set(color_codes=True)
pal = sns.color_palette("viridis", 10)
sns.set_palette('muted')

date = pd.read_excel("time-hours.xlsx")
date

date.info()

sns.pairplot(date)

name =
["index_A","index_B","index_C","index_D","Temperature_of_system1",
"Temperature_of_system2",  "Mineral_parameter1",
"Mineral_parameter2",  "Mineral_parameter3",  "Mineral_parameter4"]
date[name]

df = pd.DataFrame(date)
corr = df.corr()
plt.figure(figsize=(16,12))
sns.set_context('paper',font_scale=1.4)
sns.heatmap(corr, cmap='Blues', annot=True)

from sklearn.multioutput import MultiOutputRegressor
```

附录 3：问题一：MVSR 模型的实现（Python）

```
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.metrics import explained_variance_score,r2_score
from sklearn.metrics import mean_absolute_percentage_error

name_X = ["Temperature_of_system1",    "Temperature_of_system2",
"Mineral_parameter1",  "Mineral_parameter2",  "Mineral_parameter3",
"Mineral_parameter4"]
name_y = ["index_A","index_B","index_C","index_D"]
name =
["index_A","index_B","index_C","index_D","Temperature_of_system1",
"Temperature_of_system2",  "Mineral_parameter1",
"Mineral_parameter2",  "Mineral_parameter3",  "Mineral_parameter4"]
X = date[name_X]
y = date[name_y]

svr = SVR(epsilon=0.2,kernel='rbf')
mor = MultiOutputRegressor(svr)
mor = mor.fit(X_train, y_train)
y_pred = mor.predict(X_test)
y_test = y_test.values
mse1 = mean_squared_error(y_test[:,0], y_pred[:,0])
mse2 = mean_squared_error(y_test[:,1], y_pred[:,1])
mse3 = mean_squared_error(y_test[:,2], y_pred[:,2])
mse4 = mean_squared_error(y_test[:,3], y_pred[:,3])
print("MSE:")
print("1  " + str(mse1))
print("2  " + str(mse2))
print("3  " + str(mse3))
print("4  " + str(mse4))

mse1 = mean_absolute_error(y_test[:,0], y_pred[:,0])
mse2 = mean_absolute_error(y_test[:,1], y_pred[:,1])
mse3 = mean_absolute_error(y_test[:,2], y_pred[:,2])
mse4 = mean_absolute_error(y_test[:,3], y_pred[:,3])
# print(f'MAE for first regressor: {mae_one} - second regressor:
{mae_two}')
print("MAE:")
print("1  " + str(mse1))
print("2  " + str(mse2))
print("3  " + str(mse3))
print("4  " + str(mse4))
```

附录 3：问题一：MVSR 模型的实现（Python）

```
mse1 = explained_variance_score(y_test[:,0], y_pred[:,0])
mse2 = explained_variance_score(y_test[:,1], y_pred[:,1])
mse3 = explained_variance_score(y_test[:,2], y_pred[:,2])
mse4 = explained_variance_score(y_test[:,3], y_pred[:,3])
print("可解释的方差分数:")
print("1  " + str(mse1))
print("2  " + str(mse2))
print("3  " + str(mse3))
print("4  " + str(mse4))


mse1 = r2_score(y_test[:,0], y_pred[:,0])
mse2 = r2_score(y_test[:,1], y_pred[:,1])
mse3 = r2_score(y_test[:,2], y_pred[:,2])
mse4 = r2_score(y_test[:,3], y_pred[:,3])
print("r2_score:")
print("1  " + str(mse1))
print("2  " + str(mse2))
print("3  " + str(mse3))
print("4  " + str(mse4))

mse1 = mean_absolute_percentage_error(y_test[:,0], y_pred[:,0])
mse2 = mean_absolute_percentage_error(y_test[:,1], y_pred[:,1])
mse3 = mean_absolute_percentage_error(y_test[:,2], y_pred[:,2])
mse4 = mean_absolute_percentage_error(y_test[:,3], y_pred[:,3])
print("mean_absolute_percentage_error:")
print("1  " + str(mse1))
print("2  " + str(mse2))
print("3  " + str(mse3))
print("4  " + str(mse4))

pr = pd.read_excel("preid.xlsx")
pr

pt_val = mor.predict(pr[name_X])
pt_val
```

附录 4：问题二：随机森林、集成学习、投票法（Python）

```
    import pandas as pd
import seaborn as sns
```

附录 4：问题二：随机森林、集成学习、投票法（Python）

```
from catboost import CatBoostRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputRegressor
from sklearn.svm import SVR

sns.set(color_codes=True)
pal = sns.color_palette("viridis", 10)
sns.set_palette('muted')

date = pd.read_excel("time-hours.xlsx")

name_X = ["index_A","index_B","index_C","index_D",
"Mineral_parameter1",  "Mineral_parameter2",  "Mineral_parameter3",
"Mineral_parameter4"]
name_y = ["Temperature_of_system1",    "Temperature_of_system2"]
name =
["index_A","index_B","index_C","index_D","Temperature_of_system1",
"Temperature_of_system2",  "Mineral_parameter1",
"Mineral_parameter2",  "Mineral_parameter3",  "Mineral_parameter4"]
X = date[name_X]
y = date[name_y]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=33)

svr1 = SVR(epsilon=0.2,kernel='rbf')
svr2 =
xgb.XGBRegressor(objective='reg:squarederror',**other_params)
svr3 = RandomForestRegressor(max_depth=2, random_state=0)
params = {
    'iterations':330,
    'learning_rate':0.1,
    'depth':10,
    'loss_function':'RMSE'
}
svr4 = CatBoostRegressor(**params)

models = list()
models.append(('xg', svr2))
models.append(('svr', svr1))
models.append(('RFR', svr3))
models.append(('cat', svr4))
```

附录 4：问题二：随机森林、集成学习、投票法（Python）

```
svr = VotingRegressor(estimators=models)
mor = MultiOutputRegressor(svr)
mor = mor.fit(X_train, y_train)

y_pred = mor.predict(X_test)
y_test = y_test.values
```

附录 5：问题三：GAN 思想应用、增强数据模型（Python）

```
data = pd.read_excel("data.xlsx")
name =
["index_A","index_B","index_C","index_D","Temperature_of_system1",
"Temperature_of_system2",  "Mineral_parameter1",
"Mineral_parameter2",  "Mineral_parameter3",
"Mineral_parameter4","Process_parameter3","Process_parameter4"]
date = data[name]
name_X = ["Temperature_of_system1",    "Temperature_of_system2",
"Mineral_parameter1",  "Mineral_parameter2",  "Mineral_parameter3",
"Mineral_parameter4","Process_parameter3","Process_parameter4"]
name_y = ["index_A","index_B","index_C","index_D"]
name =
["index_A","index_B","index_C","index_D","Temperature_of_system1",
"Temperature_of_system2",  "Mineral_parameter1",
"Mineral_parameter2",  "Mineral_parameter3",  "Mineral_parameter4"]
X = date[name_X]
y = date[name_y]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=30)
other_params = {'learning_rate': 0.1, 'n_estimators': 300,
'max_depth': 5, 'min_child_weight': 1, 'seed': 0, 'subsample': 0.8,
'colsample_bytree': 0.8, 'gamma': 0, 'reg_alpha': 0, 'reg_lambda':
1}
svr2 =
xgb.XGBRegressor(objective='reg:squarederror',**other_params)
params = {
    'iterations':300,
    'learning_rate':0.1,
    'depth':10,
    'loss_function':'RMSE'
}
```

附录 5：问题三：GAN 思想应用、增强数据模型（Python）

```
svr4 = CatBoostRegressor(**params)

models = list()
models.append(('xg', svr2))
models.append(('CATBOOST', svr4))
svr = VotingRegressor(estimators=models)
mor = MultiOutputRegressor(svr)
mor = mor.fit(X_train, y_train)
y_test = y_test.values
y_pred = mor.predict(X_test)
pt_val = mor.predict(pr[name_X])

### 开始对抗
Virtual_val = mor.predict(X)
# 构建一个分类模型 A,B,C,D -> 0/1
df = pd.DataFrame(Virtual_val)
# df.to_excel("Virtual_val.xlsx")
df.columns = ['A','B','C','D']
df.to_excel("Virtual_val.xlsx")
ral = pd.read_excel("./gan.xlsx")
ral = pd.DataFrame(ral)
gan_data = df
df["is_qualified"] = ral["is_qualified"]
df = pd.DataFrame(df)
# 拼接
gan_data = pd.concat([df,ral])
name_X1 = ['A','B','C','D']
name_y1 = "is_qualified"
X1 = gan_data[name_X1]
y1 = gan_data[name_y1]
X1 = X1.values
y1 = y1.values
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1,
test_size=0.3, random_state=35)
clf = CatBoostClassifier(iterations=100,
                         depth=6,
                         learning_rate=0.01,
                         loss_function='MultiClass',
                         logging_level='Verbose')
clf.fit(X_train1,y_train1,eval_set=(X_test1, y_test1),plot=True
y_pred1 = clf.predict(X_test1)
```

附录 6：问题四：PSO 优化算法（Python）

```
import numpy as np
from sko.tools import func_transformer
from .base import SkoBase
from .operators import crossover, mutation, ranking, selection
from .operators import mutation

class PSO(SkoBase):
    def __init__(self, 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.random.uniform(low=self.lb, high=self.ub,
size=(self.pop, self.n_dim))
        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 = False
        self.record_value = {'X': [], 'V': [], 'Y': []}
        self.best_x, self.best_y = self.gbest_x, self.gbest_y  #
history reasons, will be deprecated

    def check_constraint(self, x):
        # gather all unequal constraint functions
        for constraint_func in self.constraint_ueq:
            if constraint_func(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)

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

    def cal_y(self):
        self.Y = self.func(self.X).reshape(-1, 1)
        return self.Y

    def update_pbest(self):
        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):
        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=None, N=20):
        self.max_iter = max_iter or self.max_iter
        c = 0
        for iter_num in range(self.max_iter):
            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
    fit = run
```