五一数学建模竞赛

承　诺　书

我们仔细阅读了五一数学建模竞赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与本队以外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其它公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们愿意承担由此引起的一切后果。

我们授权五一数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

参赛题号（从 A/B/C 中选择一项填写）：B

参赛队号：T1038944089

参赛组别（研究生、本科、专科、高中）：本科

所属学校（学校全称）：华中科技大学

参赛队员：　队员 1 姓名：马世拓

　　队员 2 姓名：孙萌竹

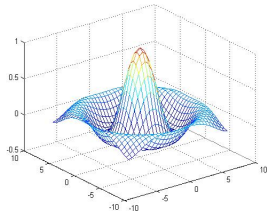
　　队员 3 姓名：飘茵

联系方式：　Email：2793055528@qq.com　联系电话：17386229686

日期：2022 年 5 月 3 日

（除本页外不允许出现学校及个人信息）

五 一 数 学 建 模 竞 赛



题 目：基于集成学习的矿石加工质量控制研究

关键词：随机森林，XGBoost，矿石加工，集成学习

摘　要：矿石加工是一个多阶段过程，提高矿石加工质量对于节能减排、“双碳”目标有着重要意义，但不同的参数控制下采用不同的温度得到的矿石产品质量也不尽相同。对此，基于提出的问题，应用**集成学习**模型进行了一定探究。

问题一是一个预测问题，将不同时间粒度的数据整合以后通过构建集成学习模型，对比了**随机森林**和 XGBoost 两个最经典的**集成学习**方法对数据切分后进行了合理预测。最终预测误差显示 XGBoost 在几类常用机器学习算法中误差最低，基本做到与原始数据走向一致。

问题二同样是一个预测问题，与问题一不同，此时以四项指标和参数为自变量预测两项温度。最终结果同样显示 XGBoost 效果最佳，R^2等于 1.00。

问题三是一个分类问题，我们按四项指标的范围筛选出合格产品进行标注，同样经过数据的粒度整合以后以温度、参数和过程数据作为自变量预测是否合格，将样本被分类为合格的概率作为合格率对问题进行了解决。最终**随机森林**和 XGBoost 的效果都非常好，做到了 100%精准预测。

问题四则在问题三的基础上，通过网格搜索的方式生成测试数据集进行测试获得合格率随两系统温度的变化关系，在温度尽可能低的情况下找到达到目标合格率的条件。结果表明在 4 月 11 日的参数下系统无法达到 99%的合格率，但 4 月 10 日可以达到 80%的合格率。

我们的方法较为精准且鲁棒，有效解决了传统拟合方法所难以解决的预测问题，并且能够高速并行计算，适合处理大规模数据，具有良好的可扩展性。有关结果均已经写入正文对应的表格中。

一、 问题重述

1.1 问题背景

在矿石加工过程中，电压、水压、温度都是影响矿石加工的重要因素，直接会影响矿石产品的质量。提高矿石加工质量，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，从而推动节能减排，助力“双碳”目标的实现。

矿石加工过程需要经过系统 I 和系统 II 两个环节，在这个过程中记录调节温度所对应的矿石产品质量的评价指标，生产技术人员通过传入调温指令，调节温度来改变产品质量。

1.2 问题提出

我们需要解决的问题如下：

1. 根据附件 1 给出的生产车间 2022-01-13 至 2022-01-22 的生产加工数据，建立数学模型，给出利用系统温度预测产品质量的方法。在给定的 2022-01-23 原矿参数和系统设定温度下，给出产品质量预测结果。

2. 根据问题 1 的结果，利用附件 1 的数据建立数学模型，估计产品目标质量所对应的系统温度。在给定的 2022-01-24 原矿参数和目标产品质量下，给出系统设定温度。

3. 建立数学模型，给出指定系统设定温度，预测矿石产品合格率的方法。在给定的 2022-04-08 和 2022-04-09 原矿参数、过程数据和系统设定温度下，给出合格率预测结果，填入表 4，并建立数学模型对给出的合格率的准确性进行评价。

4.根据问题 3 中的结果，利用附件 2 的数据，建立数学模型分析在指定合格率的条件下，设定系统温度的方法。并完成适当的敏感性分析，对结果准确性的分析，判断能否达到表 5 中给出的 2022-04-10 和 2022-04-11 产品的合格率要求，如果可以达到，给出系统设定温度，并将结果填入表 5。

二、 问题分析

2.1 问题一的分析

问题一需要在已知参数的情况下根据温度预测产品质量。这是一个典型的回归问题。但数据有一个重要的特征比较棘手：时间粒度不一致。于是首先需要进行时间粒度的对齐，将以日为频率的数据复制到小时粒度，分钟粒度的数据均按 50 分钟进行采样。随后我们可以对比多种机器学习模型进行求解，由于可用特征较少而且参数的变化不大，所以集成学习可能取得较好效果。

2.2 问题二的分析

问题二则需要根据四项指标反推温度。类比问题一，以参数和质量指标为自变量而以两项温度作为预测目标构建回归模型即可。

2.3 问题三的分析

问题三与问题一二不同，是一个合格率的预测问题。我们按照问题给定的筛选条件筛选出合格的产品并进行标注，同样以小时为单位对数据进行粒度对齐。过程数据因为以 3 小时为频率所以我们将前面两个小时进行复制处理。我们以温度、过程数据和参数作为自变量预测是否合格，将其抽象为一个分类问题。同样的，由于集成学习模型既可以处理分类也可以进行回归所以我们同样应用集成分类模型对问题进行处理。而对于合格率，我们将其抽象为样本被分类为合格的概率。

2.4 问题四的分析

问题四是对问题三的扩展，要求在给定合格率的条件下预测温度。这里我们并没有像问题二一样构建回归模型，而是按照温度的范围进行网格搜索生成测试数据，对测试数据进行合格率预测，观察在温度尽可能低的条件下何时才能达到要求的合格率。最终结果写入表格中。

三、 模型假设

针对这些问题，我们的模型假设如下：

1.假设系统温度与调温指令设定的温度相同。

2.假设每单位时间生产的产品数量相同，即生产矿石是匀速生产，合格率=合格产品数/产品总数，否则按日为粒度的合格率将除了与每小时的生产合格率有关以外还与生产速率有关。

3.假设模型参数在按日为频率更新的情况下一日内的参数数据不会发生显著变化，这将对问题产生显著影响。

四、 符号说明

表 1　各变量符号说明及解释

符号	说明
$p(x)$	x 的概率
$H(X)$	X 的信息熵
$H(X\|Y)$	按照 Y 划分的条件信息熵
$ID(X,Y)$	信息增益
$L(\Phi)$	XGBoost 的损失函数
$f_t(x)$	学习的基学习器
$\Omega(f_t(x))$	正则化项，与基学习器的深度与权值范数有关
λ	常数项
w	XGBoost 中的各项权值系数
accuracy	准确率
F1-score	F1 分数
precision	查准率
recall	查全率
TP	预测为正实际也为正
FP	预测为负实际为正
TN	预测为正实际为负
FN	预测为负实际也为负
Gini	基尼指数

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 数据的探索

我们对数据进行了插值，将四项指标与温度之间的关系绘制在图 1 所示的三维曲面中，顺序分别为指标 A、指标 B、指标 C、指标 D。

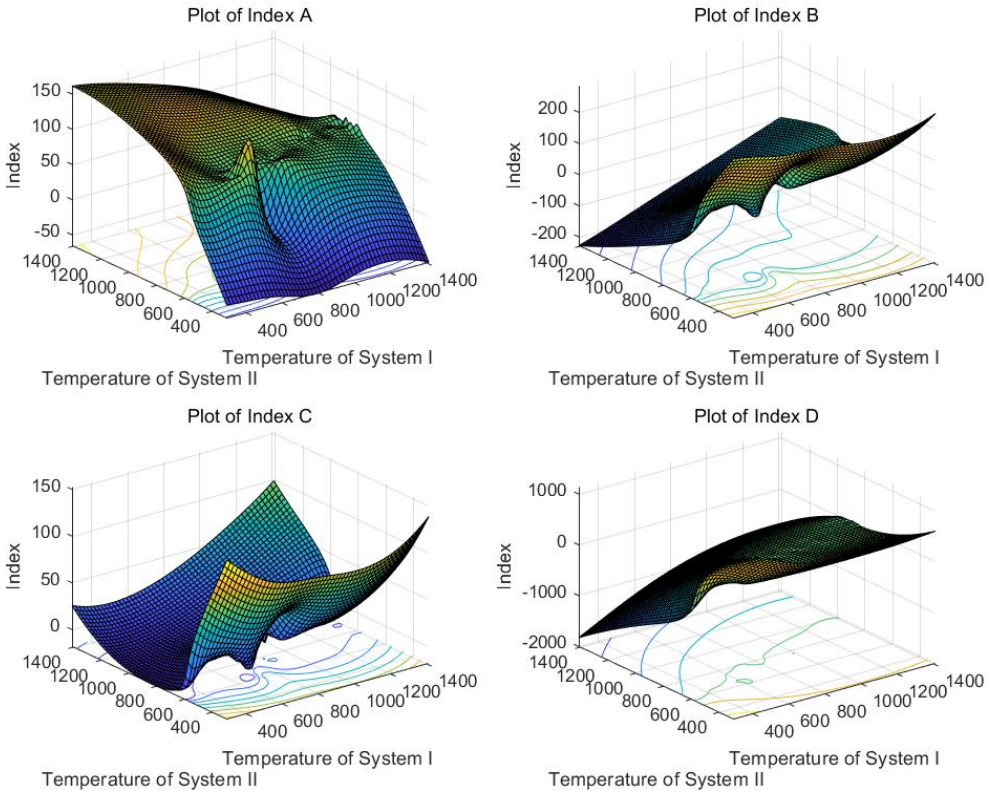


图 1. 四项指标与温度的关系曲面

从图 1 中可以看到，指标 A 与系统二的温度之间基本呈现正相关关系，相对而言，系统一的温度对于指标 A 影响的波动幅度没有系统二的影响大；指标 B 与系统二的温度之间存在基本负相关的关系；指标 C 的大小在系统二温度时在 600~800 间有一个明显的拐点；指标 D 与系统二温度间呈现负相关关系，与系统一温度基本呈现正相关趋势。

为了更深入探究四项不同指标对质量的反映，我们对四项质量指标进行了主成分分析，并将有关结果绘制在图 2 中。图 2 的四幅子图分别为方差贡献率的帕累托图、贡献矩阵热力图、成分向量分解图和四项指标随着时间的波动：

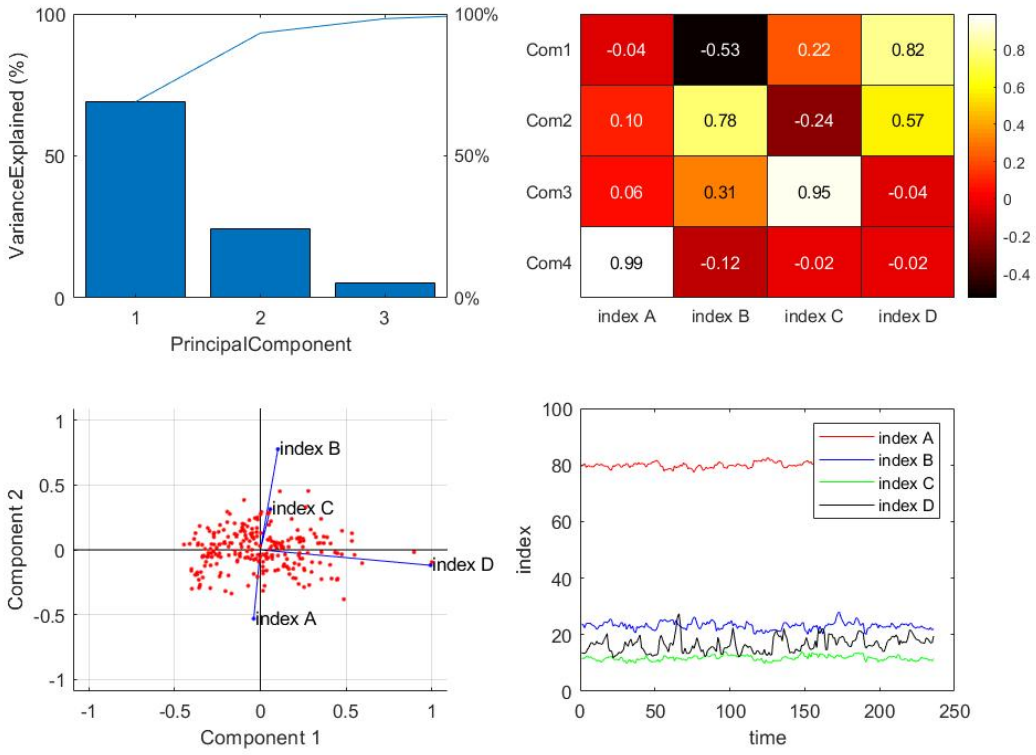


图 2. 主成分分析的结果

从图 2 中可以看出，子图 1 是方差贡献率的帕累托图，主成分 1，2 的累计方差贡献率已将超过 85%，可以提取两个主成分因子。子图 2 是贡献矩阵热力图，可以看到四个不同的指标对应四个不同主成分的权重。子图 3 的成分向量分解图表明，指标 B、C、D 与主成分因子 1 呈现正相关趋势，指标 A 与成分 1 呈负相关关系，指标 D 对于成分 1 的贡献尤为明显；指标 B、C 对于成分 2 呈现正相关关系，指标 B 的贡献明显，指标 A、D 与成分 2 呈现负相关关系，指标 A 的负向贡献较为显著。子图 4 是各个指标与时间的关系图，指标 A 在 80 附近波动，其余指标 B、C、D 均在 20 附近围绕固定值波动。其中指标 D 的波动性最明显，指标 B 的波动较大，指标 A、C 随时间波动幅度较小。

由于数据的时间粒度并不完全对应，如温度是以分钟为频率变化，质量以小时为频率变化，而参数则以天为频率变化，我们对数据先进行了对齐整理。有关时间点产生缺失的数据我们采用线性插值的方法进行了填充后再进行处理。

5.1.2 模型的建立与求解

我们以四项参数和两个系统的温度为自变量，分别对四项质量指标应用不同的机器学习模型进行了回归预测。

随机森林模型

随机森林是 Breiman[1] 提出的基于树的集成学习算法，根据特征数对每个样本选取分裂指标进而构建单棵子树去完成分类或回归任务。随机森林通过集成多性

能较弱的多个基学习器来构建一个强学习器，各个基学习器之间相互互补，降低了方差以及过拟合的风险，从而提高模型的性能。由于随机森林是基于 Bagging 策略[2]的的算法，利用 bootsrap 采样方法从原始样本中抽取多个样本，对每个样本进行决策树建模，然后综合多棵决策树的结果,通过投票得出最终预测结果，具有很高的预测准确率,对异常值和噪声具有很好的容忍度,其基学习器之间并无直接关联，于是可以很方便地进行并行化计算，且不容易出现过拟合[3]。

由于随机森林基于 Bagging 策略，其基学习器只会抽样式从服从同一分布的随机向量中对数据进行采样并学习基学习器，最终应用投票法将多个基学习器的结果投票进行输出。其学习算法如下：

Step 1. 采用 Bootstrap 策略从原始数据集中采样生成 N 个子数据集，并在每个子数据集上训练 CART 决策树[4]。

Step 2. 随机森林由 i 棵分类树构成，每棵分类树的子节点在进行分裂时随机选择分裂指标数，根据衡量指标大小选择最优分割指标进行划分。

Step 3. 重复第二步，使每棵子树都构建完成。

Step 4. 将生成子树进行投票获得最终学习器：

$$P_{rf}(X)=arg\max_{Y}\sum_{i=1}^{n}I(w(X,\theta_i)=Y)NY_c \tag{1}$$

随机森林的决策结果取决于每一棵子树的训练结果，分裂指标的选取决定了分裂标准，随机森林一般采用基尼指数 (Gini)[5]，其大小衡量了各节点混乱程度，其计算方法如下:

$$Gini(c)=1-\sum_{y=1}^{u}p(y|c)^2 \tag{2}$$

我们将模型的思想流程图绘制在图 3 中：

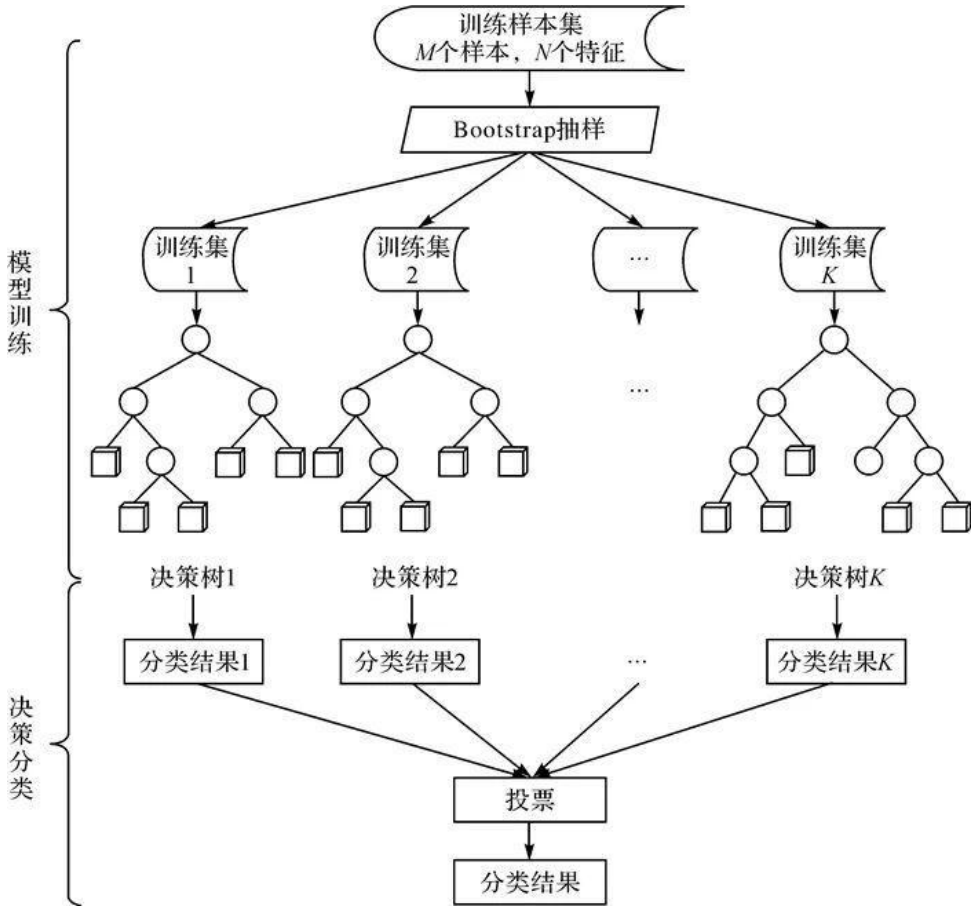


图 3. 随机森林训练流程

XGBoost 模型

由 Chen 等人提出的 XGBoost 是对梯度提升树框架(Gradient Boost Decision Tree Framework)的一种具体实现，以 CART 决策树作为基学习器，既可以用于分类问题也可以用于回归问题[6]。XGBoost 的基本形式为

$$\begin{cases} L(\phi)=\sum_i l(\hat{y}_i,y_i)+\sum_k \Omega(f_k) \\ \Omega(f)=\gamma T+\frac{1}{2}\lambda\,||\omega||^2 \end{cases} \tag{3}$$

在迭代过程中的损失函数表达式如下

$$\begin{aligned} L^{(t)} &= \sum_{i=1}^{n} l\left(y_i,\hat{y}_i^{(t-1)}+f_t(x_i)\right)+\Omega(f_t) \\ &\simeq \sum_{i=1}^{n}\left[l\left(y_i,\hat{y}^{(t-1)}\right)+g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x_i)\right]+\Omega(f_t) \end{aligned} \tag{4}$$

不同于普通的 GBDT，XGBoost 在 L 后加入了与决策树深度 T 和分类器分数 w

有关的正则化项，使得精度本就出众的模型更具有鲁棒性。此外从某种意义上讲，它本身还具有降维的效果，并且对类别失衡的数据效果较好。

XGBoost 由于是基于 Boosting 原理训练基学习器，每一步的学习器是在上一步学习器的基础上改进得到。其迭代训练算法如下所示：

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted*$(I,$ *by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

图 4. XGBoost 算法训练过程

XGBoost 使用了和 CART 回归树一样的想法，利用贪婪算法，将所有特征升序排序，然后遍历所有特征的所有特征划分点，不同的是使用的目标函数不一样。具体做法就是求分裂后的目标函数值比分裂前的目标函数的增益。而 XGBoost 虽然采用了 Boosting 系列[7]方法，但同样可以在特征级上做到并行训练，对不同特征的基尼指数进行并行化计算。

我们对比了线性回归、支持向量机[8]、决策树、xgboost 和随机森林五种算法在回归问题上的效果，按照 3:1 的比例划分训练集和测试集，将四项指标的回归曲线绘制在图 5 中：

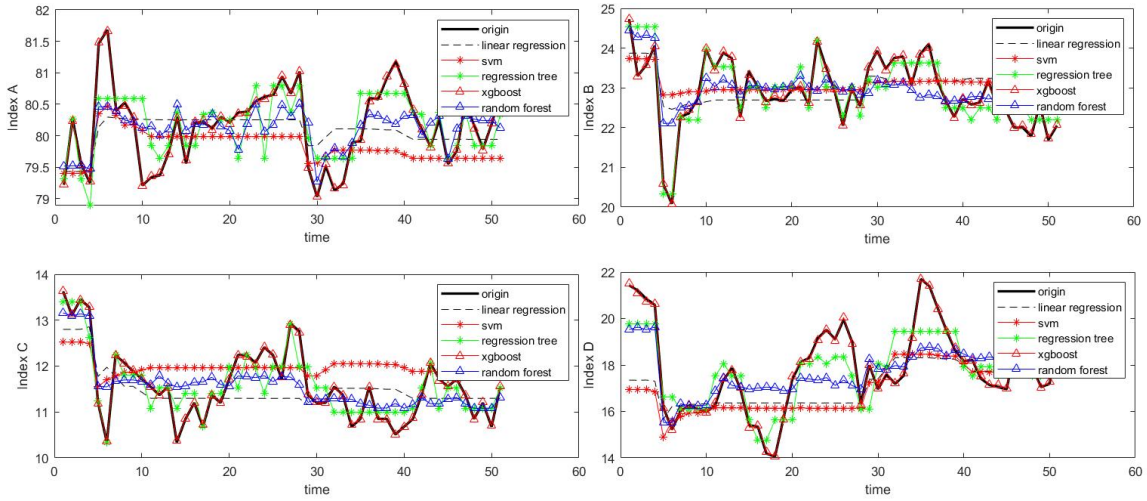


图 5. 不同算法预测质量的回归效果

从图 5 中可以看到，对于各个指标来说，xgboost 的拟合效果最好，决策树的拟合效果次之，随机森林对于指标随着时间变化的趋势预测与原始数据基本一致，而线性回归与 SVM 的拟合能力较弱。以 xgboost 为代表的集成学习算法在这一预测问题上取得了成功。

我们采用 RMSE 作为回归的衡量指标：

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5}$$

将不同模型的 RMSE 绘制在图 6 所示的条形图中：

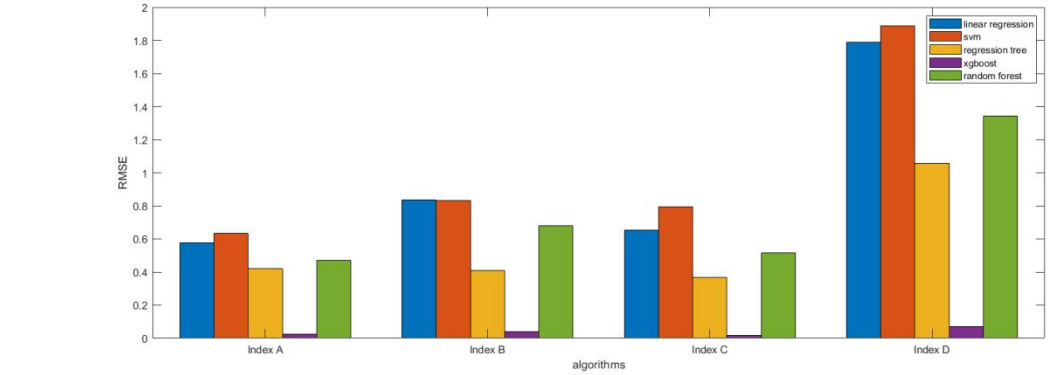
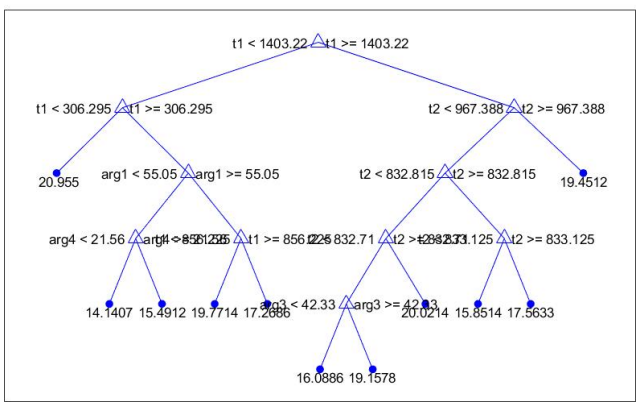


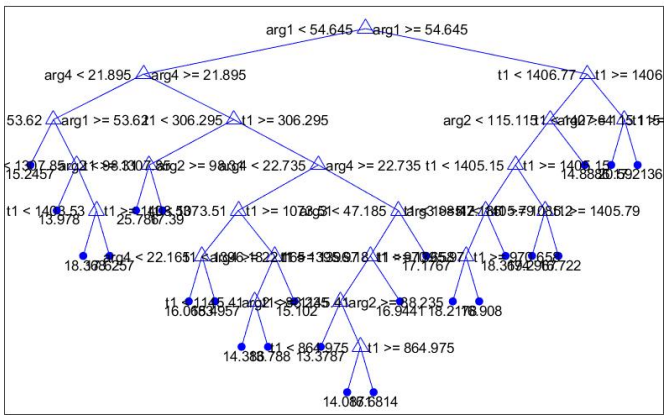
图 6. 不同算法的回归误差

图 6 的结果表明，线性回归与 SVM 在各个指标中的回归误差较为接近且误差较大，随机森林模型的误差次之，再次是决策树的回归误差，而 xgboost 在各个指标中回归误差非常小，拟合效果最好。所有模型回归误差在指标 A，B，C 差异不大，但对于指标 D 的预测误差都有显著增加。

为了深入探究两类集成学习器的结构差异，我们将 xgboost 和随机森林的基学习器结构抽样绘制在图 7 中。可以发现它们都是决策树类的集成。



(a). xgboost 基学习器的结构



(b). 随机森林基学习器的结构

图 7. 两类集成学习方法的基学习器结构

从图 7 中可以发现，由于 xgboost 算法采取的是 boosting 策略，其学习器在迭代的过程中逐渐变得复杂，所以初始情况下 xgboost 的基学习器结构会更简单，故能更快并且有很强的鲁棒性。而随机森林由于应用的是 bagging 策略，所有的基学习器之间并不互相依赖，故基学习器之间结构复杂度类似。但由于随机森林的基学习器并不互相依赖故能很容易实现并行处理。

最终我们将问题一的预测结果写入表 2 并展示如下：

表 2. 问题一的结果

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	79.83	22.57	11.70	18.03
2022-01-23	1151.75	859.77	79.11	24.04	12.32	15.57

5.2 问题二模型的建立与求解

5.2.1 模型的求解

问题二是对问题一的迁移应用。我们以四项质量指标和四个参数为自变量，而以两个系统的温度为预测目标，同样对比了不同算法的效果并绘制在图 8 中。

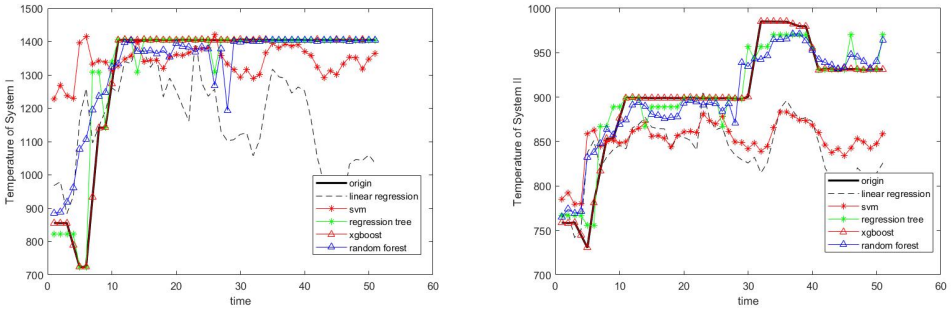


图 8. 不同算法预测温度的回归效果

从图 8 中我们可以看到，xgboost 的预测效果仍然是最好的，决策树次之。随机森林对于温度的长期预测较为准确，但对于短期预测的效果不是很好。线性回归对于短期预测的趋势较为一致，但对于长期预测效果不好。SVM 对于系统一温度的预测效果不是很好，但对于系统二温度的预测结果不好，变动趋势与原始值基本一致。这一问题同样是 xgboost 取得最优。

我们也对回归结果与原始数据的相关性做了基本分析。以系统 1 的温度为例，我们绘制了图 9 所示的曲线分析预测值与原始数据的相关性：

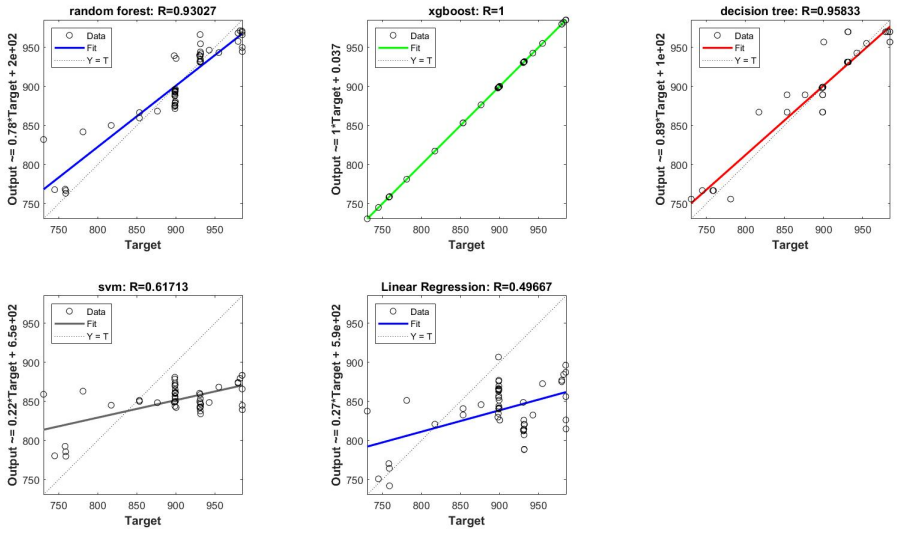


图 9. 系统温度预测值与原始值的相关性

从图 9 中我们可以看到，xgboost 预测值与原始数据拟合程度高，预测效果最好；随机森林模型与决策树模型预测效果接近；SVM 的解释程度不是很高，在高温区的预测效果不好；而线性回归则是预测效果最差的方式。

我们将问题二的结果写入表 3：

表 3. 问题二的结果

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	1103.60	798.33
2022-01-24	80.10	23.34	11.03	13.29	976.81	758.23

5.3 问题三模型的建立与求解

5.3.1 数据的探索

问题三则与问题一二不同，是一个分类问题。我们根据题目所给的条件筛选出合格产品和不合格产品。将不同的属性绘制成对矩阵图如图 10 所示：

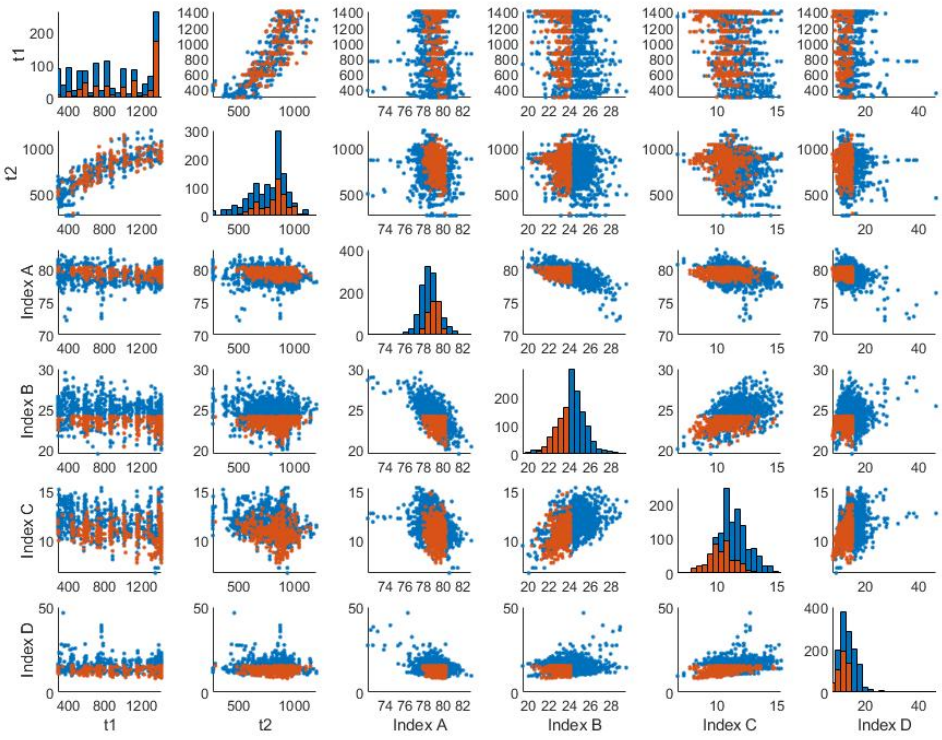


图 10. 原始数据的成对矩阵图

在图 10 中，红色点为合格产品，蓝色点为不合格产品。可以看到，指标的相关性并不算强烈。其中，B 指标与 C 指标之间、系统 I 温度和系统 II 温度之间存在一定程度的相关性，而其他指标之间近乎独立。温度与四项指标之间关系及不明显。但可以观察到两个有趣的现象：

1. 对于以温度作为坐标轴的关系图，其值总是在某一指标值上下浮动。

2. 对于指标之间的关系图，合格产品在其中的表示存在明显的截断线。这也与合格产品是基于四项指标的取值范围界定区域有关，故合格产品的分类边界必然是

平行于指标坐标轴的直线。

上述结论也更加映证了使用决策树类算法或决策树类集成学习算法将取得更好的效果。

5.3.2 模型的求解

分类问题同样可以使用问题一中的集成学习模型进行分类。而为了衡量分类的效果，我们也有一些分类指标对其进行衡量。

对于分类问题，预测值和真实值之间可以构建混淆矩阵。而基于混淆矩阵可以得到以下一些指标：

准确率即为预测值与真实值相同的样本在样本整体中占比：

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

在此基础上我们提出了查准率和查全率的概念。查准率为分类正确的正样本数 (TP) 占 预测为正样本的总样本数 (TP + FP) 的比例，其定义为：

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

而查全率为分类正确的正样本数 (TP) 占 实际的正样本总数 (TP + FN) 的比例，其定义为：

$$\mathrm{Re}call = \frac{TP}{TP + FN} \tag{8}$$

F1 分数即为查准率和查全率的调和平均：

$$F_1 = \frac{2}{\frac{1}{Precision} + \frac{1}{\mathrm{Re}call}} \tag{9}$$

理想时，查准率和查全率二者都越高越好。然而，二者实际是一对矛盾的度量。一个原因是样本存在误差，尤其是临界边缘的数据由于存在误差，导致互相渗透；另一个原因是 模型拟合能力有限，非线性模型以超平面划分样本，若扩大召回，可能导致混入更多错误样本，若提升精度，必然召回会下降。例如，若某模型很贪婪，想要预测尽可能多的样本，那么犯错的概率就会随样本数增加而提升；相反，若某模型很保守，会尽量在 “更有把握” 时才把样本预测为正样本，但会因过于保守而漏掉很多 “没有把握” 的正样本，导致 Recall 值降低。通常，只有某些简单任务才会二者都高。因此，在不同场合需根据实际需求判断哪个指标更紧要。

AUC 值以 TPR 和 FPR 为横纵坐标绘制曲线：

$$\begin{cases} TPR = \dfrac{TP}{TP + FN} \\ FPR = \dfrac{FP}{FP + TN} \end{cases} \tag{10}$$

曲线的面积即为 AUC 值。

我们同样对比了几种不同算法在这一分类问题中的效果，将数据集按照 7：3 的比例切分以后进行训练，经过测试结果评估绘制了如图 11 所示的 ROC 曲线：

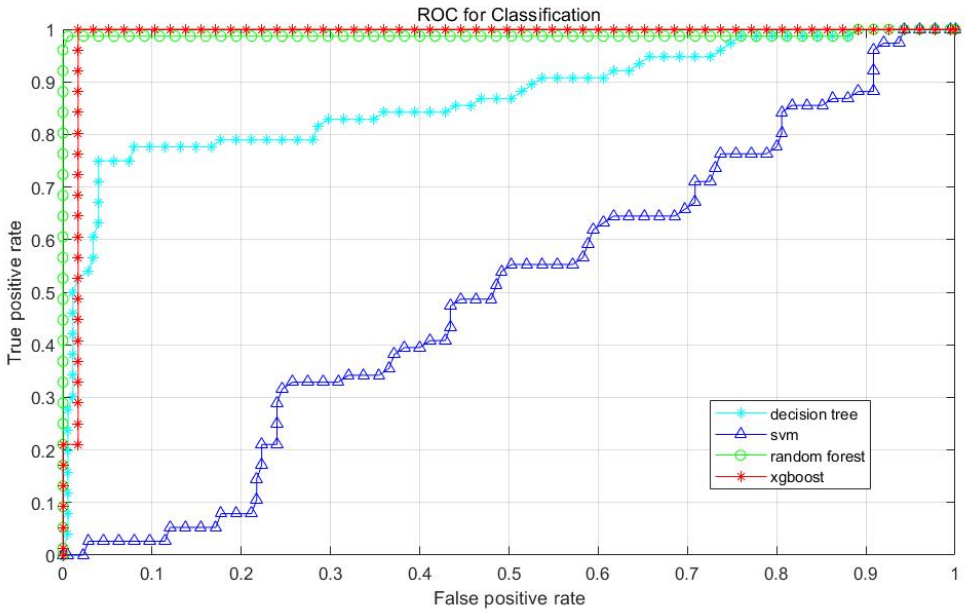


图 11. 分类的 ROC 曲线

从图 11 中我们可以看出，对于这一分类问题，集成学习的两种代表方法（XGboost 和随机森林）都取得了极大的效果。而决策树模型则显得弱一些，AUC 值更小。支持向量机在这一问题中表现极其不鲁棒，因为它将所有样本都判定为不合格，使得召回率为 0。

我们也将模型效果绘制在图 12 所示的条形图和表 4 所示的分类结果中：

表 4. 分类器的性能

算法	precision	recall	F1-score	accuracy	AUC
SVM	/	0	/	0.70	0.49
决策树	0.88	0.75	0.81	0.89	0.87
随机森林	1.00	1.00	1.00	1.00	1.00
xgboost	0.96	1.00	0.98	0.99	0.99

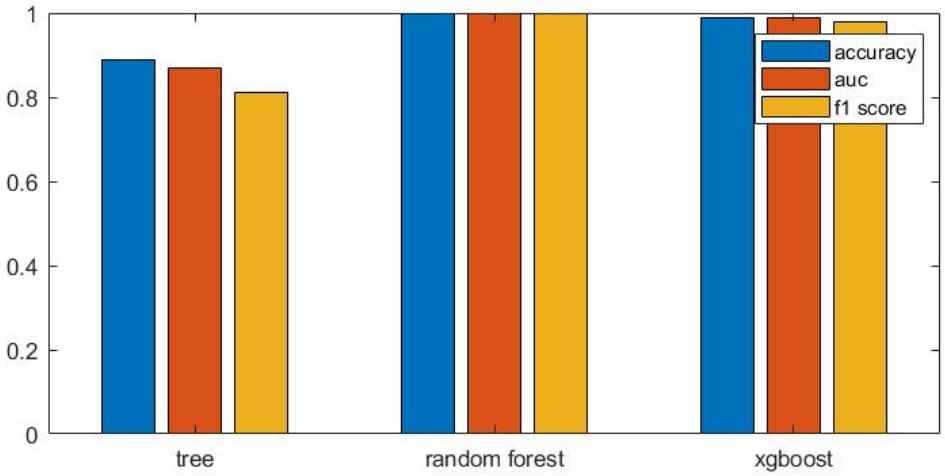
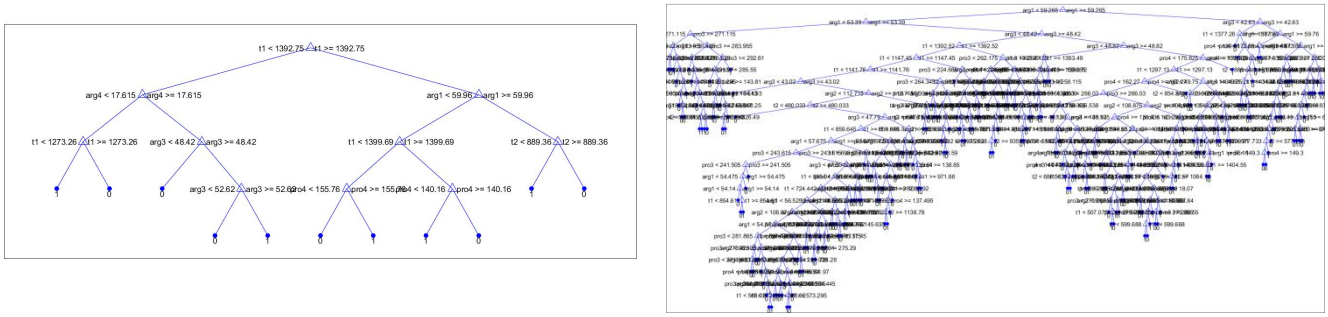


图 12. 分类器性能

从图 12 中我们可以看到，随机森林和 xgboost 算法的性能非常接近几乎都达到了 100%。这表明集成学习方法在合格分类的问题上是合适且高性能的。单一决策树在各方面性能都相较集成学习要弱一些。

同样的，我们也将两类集成学习的基学习器结构展示在图 13 中：



(a). xgboost 基学习器结构　　(b). 随机森林基学习器结构

图 13　两类集成学习分类的基学习器结构

可以看到，对分类问题而言 boosting 策略和 bagging 策略的复杂度差异更加明显。Xgboost 的基学习器是在不断迭代变化于是更加复杂，所以在初始阶段 xgboost 的分类准则是简单而明确的，容易进行可视化。而随机森林不同，由于其高度并行的机制使得其基学习器必须在较大样本空间的数据集上进行训练，而且学习器之间没有直接关联使得即使是随机森林的基学习器复杂程度也比较高。

对于合格率的转化，我们认为，由于决策树以及决策树集成类的机器学习算法是可以输出分类概率的，我们可以将分类为合格的概率作为当前时刻的合格率（数据以小时为粒度则代表这一小时生产产品的合格率）。由于假设中认为每一小时生产的产品数量是均匀的，我们也可以用 24 小时内生产产品的合格率平均值作为平均合格率代替。

最终我们基于提供数据进行预测，将问题三的结果展示在表 5 当中：

表 5. 问题三的结果

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	26.4%
2022-04-09	1010.32	874.47	82.0%

5.4 模型四的建立与求解

5.4.1 模型的建立

对于问题四，由于两类集成学习器都支持并行计算且速度较快，我们对两个系统的温度生成网格数据后进行了网格化搜索，对 4 月 10 日和 4 月 11 日不同的参数配置下进行温度与合格率的曲面绘制，如图 14 所示：

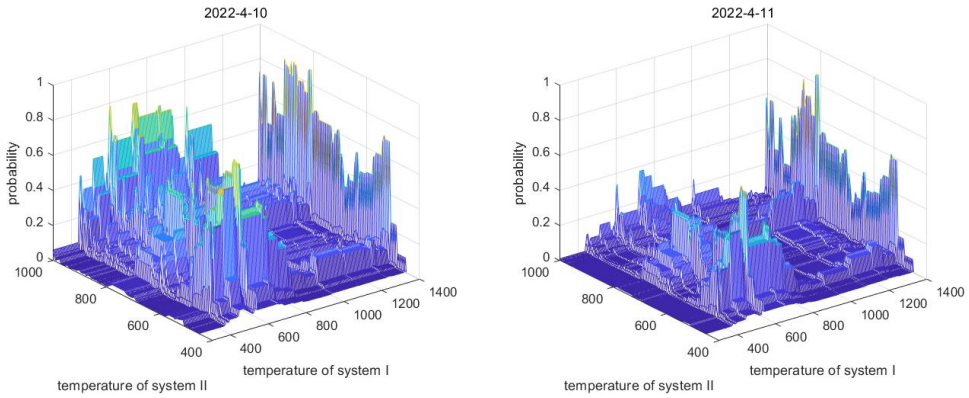


图 14. 两系统温度与合格率的变化关系

图 14 反映了系统温度和合格率变化的关系。可以看到，温度与合格率之间并不是连续的变化关系而是非常离散。但总体而言当系统 I 温度处于低温段（大概 400-800 度左右）而系统 II 温度处于中高温段（大概 700-900 度左右）或低温段（450-550 度左右）时分类为合格产品的概率更高。

我们也对模型进行了灵敏度分析。分别以参数配置和过程数据为例，保持其他量相同，仅改变参数 1 或过程 1 的条件下对合格率进行的分析。由于单条曲线没有明确的变化规律，为方便起见这里按照相同的顺序对预测数据进行了排序以后绘制曲线如图 15 所示：

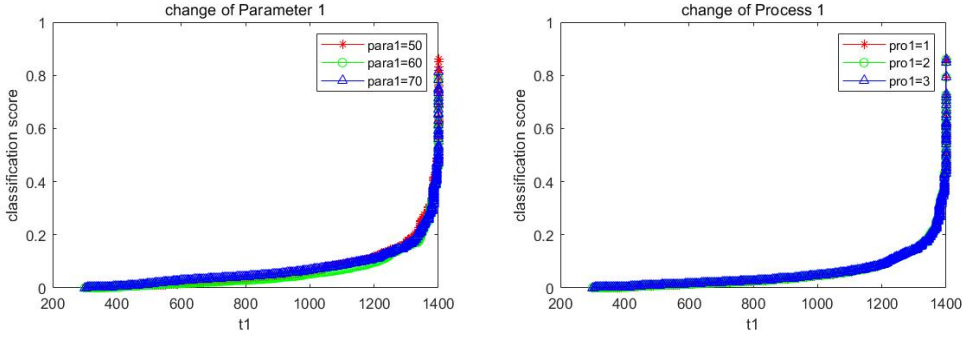


图 15. 灵敏度曲线

从图 13 中可以看出，无论是改变参数还是改变过程，三条曲线基本一致没有太大差异。因此我们认为这一模型对其他参数并不灵敏。

最终，我们将问题四的结果写入表 5。可以看到，4 月 11 日是没有办法达到 99%的合格率的。

表 5. 问题四的结果

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%	能	1395	875
2022-04-11	99%	不能	/	/

六、 模型的评价、改进与推广

6.1 模型的优点

该模型从集成学习角度出发，结合机器学习算法，保证了结果的可靠性，随后我们根据一些模型评价指标评估，认为模型在本数据集上取得了较好的效果。我们认为，我们在这一系列问题中提出的模型有以下优点：

1.使用多种机器学习算法对比，结果准确可靠，模型表现好。

2.利用集成学习进行了回归和分类两种任务，具有很强的迁移性，同时结果可以高效并行和可视化。

3.模型之间的转化合理，能够针对不同的任务场景迁移式应用类似的方法。

4.模型适用于大规模数据集，能够在大规模数据上取得良好的表现。故可以用于海量大规模工业数据的预测，并且其高效的并行机制保证可以快速处理。

6.2 模型的缺点

尽管模型整体表现良好，但我们认为还存在一些不足之处，例如：

1.对于集成学习模型进行分类，在大规模数据上的并行化速率问题和模型复杂程度还有待提高。

2.预测结果生成仍然存在一定的问题，算法时间复杂度较高并且严重依赖生成树，可以考虑从神经网络的角度入手进行进一步分析。

6.3 模型的改进

该模型除了可以使用 XGBoost 和随机森林作为学习器以外，同样属于 GBDT 框架下的 LightGBM 作为 XGBoost 的一种改进算法也可以用于尝试。

6.4 模型的推广

该问题的求解方法不仅可以用于矿石加工的工业领域，结合问题实际的简单数值处理原理纯粹但有高度的可解释性和效率，采用 XGBoost 和随机森林等集成学习模型支持高度并行化，系列模型具有高度的可移植性。

七、 参考文献

[1] Breiman L . Random forest[J]. Machine Learning, 2001, 45:5-32.
[2] Quinlan J R . Bagging, Boosting, and C4.5[C]// Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 1. 1996.
[3] 李欣海. 随机森林模型在分类与回归分析中的应用[J]. 应用昆虫学报, 2013, 50(4):8.
[4] 陈辉林, 夏道勋. 基于 CART 决策树数据挖掘算法的应用研究[J]. 煤炭技术, 2011, 30(10):3.
[5] Breiman L， Friedman J H， Olshen R A , et al. Classification and Regression Trees (CART)[J]. Biometrics, 1984, 40(3):358.
[6] Chen T， Guestrin C . XGBoost: A Scalable Tree Boosting System[C]// the 22nd ACM SIGKDD International Conference. ACM, 2016.
[7] Friedman J， Hastie T， Tibshirani R . Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)[J]. Annals of Statistics, 2000, 28(2):337-374.
[8] Chang C C， Lin C J . LIBSVM: A library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology, 2007, 2(3, article 27).

八、 附录

需要搭建 matlab 的 xgboost 环境

部分代码如下：

附录 1

介绍：question 1 其他算法.m

```
x=[t1,t2,arg1,arg2,arg3,arg4];
rmse_linear=[];
rmse_svm=[];
rmse_tree=[];
rmse_xgb=[];
rmse_rf=[];

subplot(2,2,1);
y=tar1;
plot(y(length(y)-50:length(y)),'k-','linewidth',2)

%% linear
newx=[ones(length(t1),1),x];
[b,bint,r,rint,stats] = regress(y,newx);
pred=newx*b;
hold on;plot(pred(length(y)-50:length(y)),'k--');
rmse_linear=[rmse_linear,
sqrt(mean((pred(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];

%% svm
svmmdl=fitrsvm(x,y,'PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4'});
pred_svm=predict(svmmdl,x);
hold on;plot(pred_svm(length(y)-50:length(y)),'r*-');
rmse_svm=[rmse_svm,
sqrt(mean((pred_svm(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];

%% tree
treemdl=fitrtree(x,y,'PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4'});
pred_tree=predict(treemdl,x);
hold on;plot(pred_tree(length(y)-50:length(y)),'g*-');
rmse_tree=[rmse_tree,
sqrt(mean((pred_tree(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];

%% gbdt
boostmdl=fitrensemble(x,y,'Method','LSBoost','PredictorNames',{'t1','t2','arg1','arg2','arg
3','arg4'});
pred_boost=predict(boostmdl,x);
```

```
hold on;plot(pred_boost(length(y)-50:length(y)),'r^-');
rmse_xgb=[rmse_xgb,
sqrt(mean((pred_boost(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];

%% Random Forest
bagmdl=fitrensemble(x,y,'Method','Bag','PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4
'});
pred_bag=predict(bagmdl,x);
hold on;plot(pred_bag(length(y)-50:length(y)),'b^-');
rmse_rf=[rmse_rf,
sqrt(mean((pred_bag(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%%  设置
xlabel("time")
ylabel("Index A")
legend("origin","linear regression","svm","regression tree","xgboost","random forest")


subplot(2,2,2);
y=tar2;
plot(y(length(y)-50:length(y)),'k-','linewidth',2)

%% linear
newx=[ones(length(t1),1),x];
[b,bint,r,rint,stats] = regress(y,newx);
pred=newx*b;
hold on;plot(pred(length(y)-50:length(y)),'k--');
rmse_linear=[rmse_linear,
sqrt(mean((pred(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];

%% svm
svmmdl=fitrsvm(x,y,'PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4'});
pred_svm=predict(svmmdl,x);
hold on;plot(pred_svm(length(y)-50:length(y)),'r*-');
rmse_svm=[rmse_svm,
sqrt(mean((pred_svm(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% tree
treemdl=fitrtree(x,y,'PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4'});
pred_tree=predict(treemdl,x);
hold on;plot(pred_tree(length(y)-50:length(y)),'g*-');
rmse_tree=[rmse_tree,
```

```
sqrt(mean((pred_tree(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% gbdt
boostmdl=fitrensemble(x,y,'Method','LSBoost','PredictorNames',{'t1','t2','arg1','arg2','arg
3','arg4'});
pred_boost=predict(boostmdl,x);
hold on;plot(pred_boost(length(y)-50:length(y)),'r^-');
rmse_xgb=[rmse_xgb,
sqrt(mean((pred_boost(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% Random Forest
bagmdl=fitrensemble(x,y,'Method','Bag','PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4
'});
pred_bag=predict(bagmdl,x);
hold on;plot(pred_bag(length(y)-50:length(y)),'b^-');
rmse_rf=[rmse_rf,
sqrt(mean((pred_bag(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%%  设置
xlabel("time")
ylabel("Index B")
legend("origin","linear regression","svm","regression tree","xgboost","random forest")


subplot(2,2,3)
y=tar3;
plot(y(length(y)-50:length(y)),'k-','linewidth',2)

%% linear
newx=[ones(length(t1),1),x];
[b,bint,r,rint,stats] = regress(y,newx);
pred=newx*b;
hold on;plot(pred(length(y)-50:length(y)),'k--');
rmse_linear=[rmse_linear,
sqrt(mean((pred(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% svm
svmmdl=fitrsvm(x,y,'PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4'});
pred_svm=predict(svmmdl,x);
hold on;plot(pred_svm(length(y)-50:length(y)),'r*-');
```

```
rmse_svm=[rmse_svm,
sqrt(mean((pred_svm(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% tree
treemdl=fitrtree(x,y,'PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4'});
pred_tree=predict(treemdl,x);
hold on;plot(pred_tree(length(y)-50:length(y)),'g*-');
rmse_tree=[rmse_tree,
sqrt(mean((pred_tree(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% gbdt
boostmdl=fitrensemble(x,y,'Method','LSBoost','PredictorNames',{'t1','t2','arg1','arg2','arg
3','arg4'});
pred_boost=predict(boostmdl,x);
hold on;plot(pred_boost(length(y)-50:length(y)),'r^-');
rmse_xgb=[rmse_xgb,
sqrt(mean((pred_boost(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% Random Forest
bagmdl=fitrensemble(x,y,'Method','Bag','PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4
'});
pred_bag=predict(bagmdl,x);
hold on;plot(pred_bag(length(y)-50:length(y)),'b^-');
rmse_rf=[rmse_rf,
sqrt(mean((pred_bag(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%%  设置
xlabel("time")
ylabel("Index C")
legend("origin","linear regression","svm","regression tree","xgboost","random forest")


subplot(2,2,4)
y=tar4;
plot(y(length(y)-50:length(y)),'k-','linewidth',2)

%% linear
newx=[ones(length(t1),1),x];
[b,bint,r,rint,stats] = regress(y,newx);
pred=newx*b;
```

```
hold on;plot(pred(length(y)-50:length(y)),'k--');
rmse_linear=[rmse_linear,
sqrt(mean((pred(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% svm
svmmdl=fitrsvm(x,y,'PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4'});
pred_svm=predict(svmmdl,x);
hold on;plot(pred_svm(length(y)-50:length(y)),'r*-');
rmse_svm=[rmse_svm,
sqrt(mean((pred_svm(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% tree
treemdl=fitrtree(x,y,'PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4'});
pred_tree=predict(treemdl,x);
hold on;plot(pred_tree(length(y)-50:length(y)),'g*-');
rmse_tree=[rmse_tree,
sqrt(mean((pred_tree(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% gbdt
boostmdl=fitrensemble(x,y,'Method','LSBoost','PredictorNames',{'t1','t2','arg1','arg2','arg
3','arg4'});
pred_boost=predict(boostmdl,x);
hold on;plot(pred_boost(length(y)-50:length(y)),'r^-');
rmse_xgb=[rmse_xgb,
sqrt(mean((pred_boost(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];


%% Random Forest
bagmdl=fitrensemble(x,y,'Method','Bag','PredictorNames',{'t1','t2','arg1','arg2','arg3','arg4
'});
pred_bag=predict(bagmdl,x);
hold on;plot(pred_bag(length(y)-50:length(y)),'b^-');
rmse_rf=[rmse_rf,
sqrt(mean((pred_bag(length(y)-50:length(y))-y(length(y)-50:length(y))).^2))];



%%  设置
xlabel("time")
ylabel("Index D")
legend("origin","linear regression","svm","regression tree","xgboost","random forest")
```

```
f=figure;
bar([rmse_linear;rmse_svm;rmse_tree;rmse_xgb;rmse_rf]',1)
legend("linear regression","svm","regression tree","xgboost","random forest")
xlabel("algorithms")
ylabel("RMSE")
set(gca,'XTick',1:4,'XTickLabel',{"Index A","Index B","Index C","Index D"})
```