队伍编号	TFB396SXJM
赛题选择	A

基于双树复小波变换和机器学习的故障智能诊断研究

摘要

随着计算机技术和人工智能科学的发展，基于机器学习的故障智能诊断方法成为从业者的新型决策工具，该技术能够不仅能够降低原始数据的环境噪声或异常数据影响，提取可靠的波形特征判据，而且还能达到较高的准确率。

针对问题一：本文选择了双树复小波变换去噪方法，首先采用平移法构造双树复小波高低通滤波器，对原始信号进行 3 层双树复小波分解，得到高低频信号分量；然后采用最小极大方差软阈值方法对分解信号进行降噪处理；最后对降噪信号进行重构。选取了其中 2 条数据分析，并对比了基于 db3、db6、demy 小波基的小波变换。双树复小波变换算法的均方误差 MSE 平均减少约 16%、和平方误差 SSE 平均减少约 15%、均方根误差 RMSE 平均减少约 9%、平均绝对误差平均减少 10%、信噪比 SNR 平均提高约 0.6dB、决定系数 R2 平均提高约 10%。

针对问题二：本文直接提取信号的时域特征以及采用快速傅里叶变换算法提取频域特征，共计 28 个。包括均值、标准差、方根幅值、均方根、峰值、峭度、峰值因子、裕度因子、波形因数、脉冲指数等 15 个时域特征，以及频谱均值、频谱均方根、均方根频率、标准差、特征频率、平均频率、均方频率、重心频率、频率方差等 13 个频域特征。

针对问题三：利用问题一的去噪算法对数据信号进行预处理，再用问题二中的计算方法对数据集做特征工程，采用 K-Means++聚类算法聚类为两类，用测试集评价聚类的效果，还对比了 K-Means、Mini Batch K-Means 和 Birch 三种聚类算法，准确率均为 90%以下，K-Means++聚类效果最佳，准确率达到 92.8%、召回率 91.4%、耗时 91 毫秒、轮廓系数 0.115、F 值 0.921、CH 分数 10.450。

针对问题四：利用问题一的降噪模型预处理后合并成一个数据集，设定两种不同的标签，并以问题二的提取特征的方式，对数据集进行构造特征工程，再采用有监督学习算法 XGBoost 进行二分类学习，并对比了 LightGBM 和 CatBoost 两种集成学习的分类模型。XGBoost 和 CatBoost 准确率均为 100%，LightGBM 准确率 70%，XGBoost 在时间上比 CatBoost 快约 225%。

针对问题五：将数据信号分为两类，经过降噪和未经过降噪的数据，分别应用于问题三的聚类模型和问题四的分类模型，经过实验验证分析，降噪过后的数据在聚类算法中准确率提高了约 1.5%、轮廓系数约 4.8%，说明了降噪提高了信号特征的表示能力。

关键词：故障诊断；机器学习；双树复小波变换；K-Means++聚类；XGBoost 分类

目录

一、 问题重述

1.1 问题背景

仪器设备故障诊断技术是一种机器在运行过程通过技术手段去掌握机器的状态，确定其整体或局部正常或异常，早期发现故障及其原因，并能预报故障发展趋势的技术。仪器故障按照来源可分为外部型和内部型，其中外部型故障的产生多为静电放射、电磁辐射、雷暴天气、空气湿度过大等导致的电路损坏或传感器失灵，内部型故障多为齿轮破裂、电机短路等。油液监测、振动监测、噪声监测、性能趋势分析和无损探伤等为其主要的诊断技术方式。随着计算机技术和人工智能科学的发展，基于机器学习或深度学习的故障智能诊断方法成为从业者的新型决策工具，该技术能够不仅能够降低原始数据的环境噪声或异常数据影响，提取可靠的波形特征判据，而且还能达到较高的准确率。

1.2 研究意义

制造业是国民经济的主体，是立国之本、兴国之器、强国之基。18 世纪中叶开启工业文明以来，世界强国的兴衰史和中华民族的奋斗史一再证明，没有强大的制造业，就没有国家和民族的强盛。打造具有国际竞争力的制造业，是我国提升综合国力、保障国家安全、建设世界强国的必由之路。

加快发展智能制造，对于推进我国制造业供给侧结构性改革，培育经济增长新动能，构建新型制造体系，促进制造业向中高端迈进、实现制造强国具有重大战略意义。

机械故障诊断与智能制造的发展是相互联系且不可分割的。《智能制造发展规划(2016-2020 年）》中将智能制造装备与故障诊断均作为智能制造的重点发展方向，可见二者具有极其紧密的联系，因此，研究机械装备故障诊断对智能制造装备的设计和智能制造的发展均具有非常重要的意义[1]。

1.3 问题重述

基于以上背景，以及附件给出的数据，通过数据分析与建模的方法、选择或改进机器学习方法，设计一系列的仿真实验，讨论和分析以下的问题：

（1）从附件一和附件二中各选一条原始数据，采用去噪方法进行去噪处理，并除了以均方误差 MSE、和平方误差 SSE、均方根误差 RMSE 作为评价指标，还需要选择另外三种作为评价标准来评价去噪的效果。并将实验结果数据填入表 1-1 和 1-2。

（2）为了后续故障智能检测模型的输入，将附件一和附件二的所有数据信号进行特征提取，并将提取的特征填入表 2-1 和 2-2 中。

（3）根据问题二提取的特征，将附件一和附件二合并成一个数据集，采用无监督学习或者半监督学习算法进行二分类，并采用准确率、召回率、耗时以及其他三种自己选择的评价指标作为分类的评价标准，并将实验数据填入表 3 中。并确保使用的预测方法预测准确率在 90%以上，准确率标准差在 10 以内。

（4）根据问题二提取的特征，将附件一和附件二的合成一个数据集，采用有监督学习的算法进行二分类，并采用准确率、召回率、耗时以及其他三种自己选择的评价指标作为分类的评价标准，并将实验数据填入表 3 中。并确保使用的预测方法预测准确率在 95%以上，准确率标准差在 5 以内。

（5）请对比分析信号去噪前后的信号特征在信号分类实验中的区别和影响。

二、 问题分析

2.1 问题一分析

为了信号的平滑去噪，常用的经典平滑算法有滑动平均法、Savitzky-Golay 法、单独处理离群值法、FIR 滤波器、IIR 滤波器、小波去噪方法。其中小波方法是最常用的，在实验中可以选择不同的小波基以及选择不同的阈值进行对比分析，小波基的家族中有 db、sym、coif、bior、rbio、dmey、gaus、mexh 等[2]，阈值选择方法有无偏风险估计阈值、极大极小阈值、固定阈值、启发式阈值。以以均方误差 MSE、和平方误差 SSE、均方根误差 RMSE、信噪比 SNR、平均绝对误差、决定系数 R2 为评价指标，对比分析出评价指标最好的小波去噪模型，作为最终的算法模型。

2.2 问题二分析

信号特征的提取往往都是用最简单有效的参数表示信号中的信息，针对不同模型的不同输入维度，需要确定特征的维度，根据附件表 1 可知，只需要提取为每个特征提取一个值即可。首先利用问题一的去噪算法，将附件一和附件二的原始数据去噪处理，得到时间域的一维信号，再计算去噪数据信号的均值，方差，均方根，峰值因子，峭度系数，波形因子，裕度因子、脉冲因子。也可以利用小波方法提取的系数，小波滤波后的特征频率等等特征。

2.3 问题三分析

首先将附件一和附件二的利用问题一的去噪模型预处理后合并成一个数据集，并以问题二提取特征的方式，对数据集进行构造特征工程。再采用无监督学习算法进行二分类，比如 K-Means、K-Means++、MinibatchK-Means、Birch 以及其他改进的 K-Means 聚类算法。最后选择准确率、召回率、耗时、轮廓系数、F 值、CH 分数作为评价标准。

2.4 问题四分析

首先将附件一和附件二的利用问题一的去噪模型预处理后合并成一个数据集，并标准标签，附件一的数据编码为 0，附件二的编码为 1，并以问题二提取特征的方式，对数据集进行构造特征工程。再采用有监督学习算法进行二分类，比如 XGBoost、LightGBM、SVM、随机森林和决策树等分类算法。最后选择准确率、召回率、耗时、F 值、AUC 分数、精准率作为评价标准。

2.5 问题五分析

对比分析信号去噪前后的数据信号，对信号问题四分类模型的影响，并以准确率、召回率、耗时、F 值、AUC 分数、精准率作为评价标准进行讨论分析。

三、 模型的假设

1. 假设题目采集到大部分数据的采集准确，并且标签记录无误；
2. 假设所采集信号存在一定噪声，并对整体影响有限。即选择的特征参数集能够反映原数据的主要特征，又能够有比较理想去噪效果。
3. 假设 A 型故障和 B 型故障在信号上有较为明显的区别。能够进行无监督或者有监督的分类。

四、 符号说明

本文公式中变量解释如表 1 所示。

表 1 符号说明

符号	符号说明	符号	符号说明
w	小波系数	y_i	原始信号
acc	精确率	$\widehat{y_i}$	去噪后信号
$F(t)$	实测信号	$R2$	决定系数
MSE	均方误差	$\bar{x}$	均值
SSE	平方误差	$\lvert\bar{x}\rvert$	绝对平均值
$RMSE$	均方根误差	δ	方差
R_2	决定系数	x_p	峰值
SNR	信噪比	W	波形指标
MAE	平均绝对误差	I	峰值指标
$MAPE$	平均绝对百分比误差	L	脉冲指标
s_i	轮廓系数	S	裕度指标
F_1	精确率和召回率的一种加权平均	x_{max}	最大值
$Recall$	召回率	x_{min}	最小值
$Precise$	精准率	x_{ms}	均方根值
σ_x	标准差	c_i	聚类中心

五、 基于改进小波变换的降噪处理

5.1 模型的建立

小波变换阈值去噪法是 Johnstone 和 Donoho 教授于 1992 年首次提出的。它是一种非线性去噪方法，在最小均方误差意义下能达近似最优，具有实现最简单、计算量最小的特点。基本原理是：正交小波分解具有时-频局部分解的能力，在进行信号处理时，小波分量表现的幅度较大，与噪声在高频部分的均匀变现正好形成明显的对比。经小波分解后，幅值比较大的小波系数绝大多数是有用信号，而幅值较小的系数一般都是噪声，即可以认为有用信号的小波变换系数要大于噪声的小波变换系数。阈值去噪法就是找到一个合适的阈值，保留大于阈值的小波系数，将小于阈值的小波系数做相应的处理，然后，根据处理后的小波系数还原出有用信号。令实测信号为 $F(t)=s(t)+n(t)$，其中 $s(t)$ 表示有纯信号，$n(t)$ 表示噪声。进行正交小波变换后，能够最大程度的去除了信号 $F(t)$ 的相关性，将大部分能量集中在少数的、幅度相对较大的小波系数上。而噪声 $n(t)$ 经过小波变换后将分布在各个尺度下的所有时间轴上，且幅度不是很大。利用这一原理，在小波变换的各个尺度上将噪声的小波系数最大程度地减小，然后利用处理后的小波系数进行信号重构，从而达到抑制噪声的目的。

阈值去噪方法的具体步骤如下：

一般来说，一维信号的阈值消噪过程可分为三个步骤进行:

第一步：对测量信号 $F(t)$ 进行正交小波变换。选定一个正交小波和分解层数 N，对信号 $F(t)$ 进行 N 层小波分解。

第二步：对测量信号的小波变换系数进行非线性阈值处理。对第 1 到第 N 层的每一层高频系数，通过阈值函数进行处理，每层的低频系数不做处理[3]。

第三步：对处理后的小波系数进行重构。根据小波分解的第 N 层的低频系数和经过处理后的第 1 层到第 N 层的高频系数进行信号重构，从而得到原始信号的估计值。

上面去噪工作成功与否关键在于如何选取阈值和如何进行阈值的量化处理。通常常用的阈值函数有两种：硬阈值函数，软阈值函数如式（1）、式（2）所示。

硬阈值处理函数：

$$\sigma_{\lambda}^{H}(w)=\begin{cases} w, |w|\geq\lambda \\ 0, |w|<\lambda \end{cases} \tag{1}$$

软阈值处理函数：

$$\sigma_{\lambda}^{s}(w)=\begin{cases} [\mathrm{sgn}(w)(|w|-\lambda)], |w|\geq\lambda \\ 0, |w|<\lambda \end{cases} \tag{2}$$

上面两式中，w 为小波系数，λ 为选定的阈值。

图 1 显示了两种阈值处理函数的处理方法：

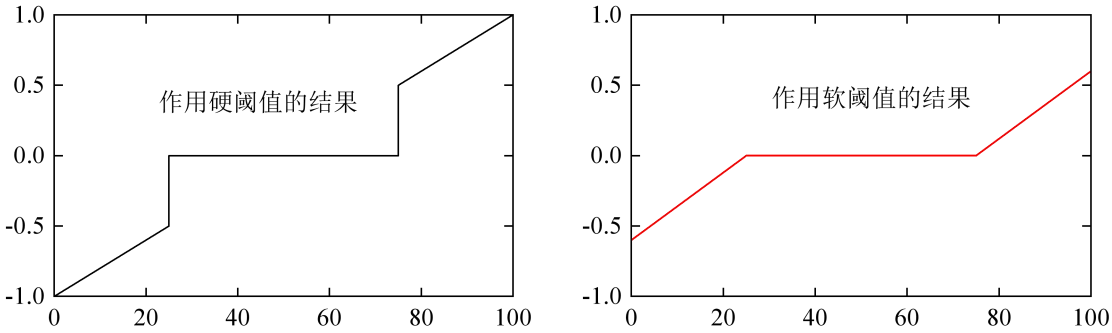

图 1 软阈值函数与硬阈值函数的处理图

从图 1 中，可以看出：硬阈值处理函数在λ处不连续，这会引起去噪后的信号会在λ处产生振荡，但是硬阈值函数在均方误差意义上优于软阈值处理函数。另外，软阈值估计得到的小波系数的整体连续性好，去噪后信号平滑一些，但是也会丢失掉某些特征。所以，在多测量信号进行去噪操作时，要视具体情况选择哪种处理函数。在阈值处理函数中，阈值λ的选取直接影响到去噪的效果，小波变换阈值去噪法的阈值的选取形式有四种：通用阈值规则、最小极大方差阈值、无偏似然估计（SURE）规则、启发式阈值规则。

（1）通用阈值规则

阈值的选取公式为：

$$\lambda = \sigma\sqrt{2\ln N} \tag{3}$$

N为实际测量的信号，σ为附加噪声信号的标准差即为对信号分解出的第一级小波系数取绝对值后再取中值。通用阈值规则在软阈值处理函数中能够得到很好的降噪效果。

（2）无偏似然估计规则

无偏似然估计规则是一种软件阈值估计器，是一种基于 Stain 的无偏似然估计（二次方程）原理的自适应阈值选择。对一个给定的阈值λ，先找到它的似然估计，然后再将其最小化，从而得到所选的阈值，具体的阈值选取规则为：

令信号$x(t)$为一个离散的时间序列，$t=1,2,\cdots,n$，再令$y(t)$为$|x(t)|$的升序序列，阈值的计算公式如下：

$$r(t) = \frac{n-2t+\sum_{i=1}^{t} y(i)+(n-t)y(t)}{n} \tag{4}$$

$$\lambda = \sqrt{\min(r(t))} \tag{5}$$

（3）启发式阈值规则

启发式阈值规则是无偏似然估计规则和通用阈值规则的折中形式。当信噪比较大时，采用无偏似然估计规则，而当信噪比小的时候，采用固定阈值规则。阈值按以下的公式计算：

$$eta = \frac{\| x \|^2 - n}{n} \tag{6}$$

$$crit = \frac{[\log(n)/\log 2]^{1.5}}{\sqrt{n}} \tag{7}$$

$$\lambda = \begin{cases} \lambda_1, \ \text{eta} < crit \\ \min(\lambda_1, \lambda_2), \ eta \geq crit \end{cases} \tag{8}$$

其中，n为信号$x(t)$的长度，λ_1为通用阈值规则得到的阈值，λ_2为无偏似然估计规则得到的阈值。

5.2 模型的求解

在对信号进行小波分解时需要选择合适的小波基，由于没有任何一种小波基可以对不同类型的信号达到最优的分解效果，因此，如何选择小波基成为小波分解的一个重点。针对现实中的信号，小波基的选择一般要考虑以下几个因素，包括支撑长度、对称性、消失矩、正则性、相似性。针对附件的一维信号，本文对比了基于双树复小波变换的去噪处理[4-6]、基于 db 基的小波变换去噪处理、基于 bior 基的小波变换去噪处理和基于 rbio 基的小波变换去噪处理。基于双树复小波变换的信号处理流程如图 2 所示。首先，确定所用小波基函数及小波分解层数，对检测信号进行分解，然后选择合适的作用阈值方法并确定阈值，对分解信号进行去噪处理;最后,对信号进行重构，得到降噪后的信号。

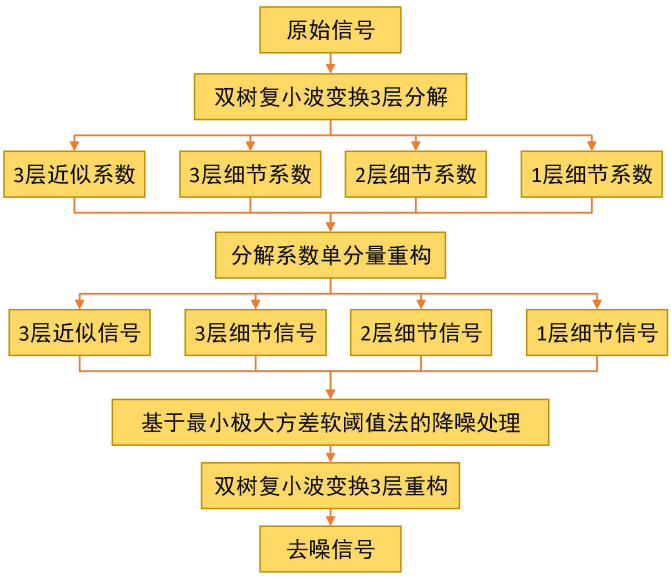

图 2 基于双树复小波变换的信号去噪处理流程

进行双树复小波分解时,若分解层数过少，则降噪效果较差，但过多的分解层数会导致计算量增大[7]。本文采用 3 层分解。采用 Q 平移法构造双树复小波变换高低通滤波器，得到高低频信号分量[8]。作用阈值在小波变换过程中具有重要意义。与硬阈值相比，采用软阈值降噪后，信号特征不会改变[9]，因此选用软阈值对信号进行降噪处理，并采用最小极大方差软阈值模型确定阈值。完成分量信号降噪处理后,对其进行重构，得到降噪后的原始信号。

5.3 评价指标

为了比较降噪效果，引入了以均方误差MSE、和平方误差SSE、均方根误差$RMSE$、决定系数$R2$、信噪比SNR、平均绝对误差MAE、平均绝对百分比误差$MAPE$以及为评价标准，对不同的降噪效果进行量化和比较，它们的定义如下：

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - \hat{y}_i\right)^2 \tag{9}$$

$$SSE = \sum_{i=1}^{n} w_i \left(y_i - \hat{y}_i\right)^2 \tag{10}$$

$$RMSE=\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i-\hat{y}_i)^2} \tag{11}$$

$$R^2=1-\frac{\sum_{i=1}^{m}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{m}(y_i-\hat{y}_i)^2} \tag{12}$$

$$SNR=10\times\log\left(\frac{\sum_{i-1}^{n}y_i^2}{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}\right) \tag{13}$$

$$MAE=\frac{1}{m}\sum_{i=1}^{m}\left|(y_i-\hat{y}_i)\right| \tag{14}$$

$$MAPE=\frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i-\hat{y}_i}{y_i}\right| \tag{15}$$

式中，y_i表示原始信号，$\hat{y}_i$表示经去噪后的信号。对于同一个含噪信号，经过处理后信号的信噪比SNR和决定系数$R2$越大，均方误差约小、和平方误差越小、均方根误差越小、平均绝对误差越小、平均绝对百分比误差越小，则降噪效果就越好。

5.4 实验结果分析

首先本文选取了第附件一中的 4.txt 和附件二中的 99.txt 作为分析样本，原始信号和去噪信号如图 3 中所示，可以看出原始信号中含有大量的噪声，在这种信号中，难以提取有效的特征。

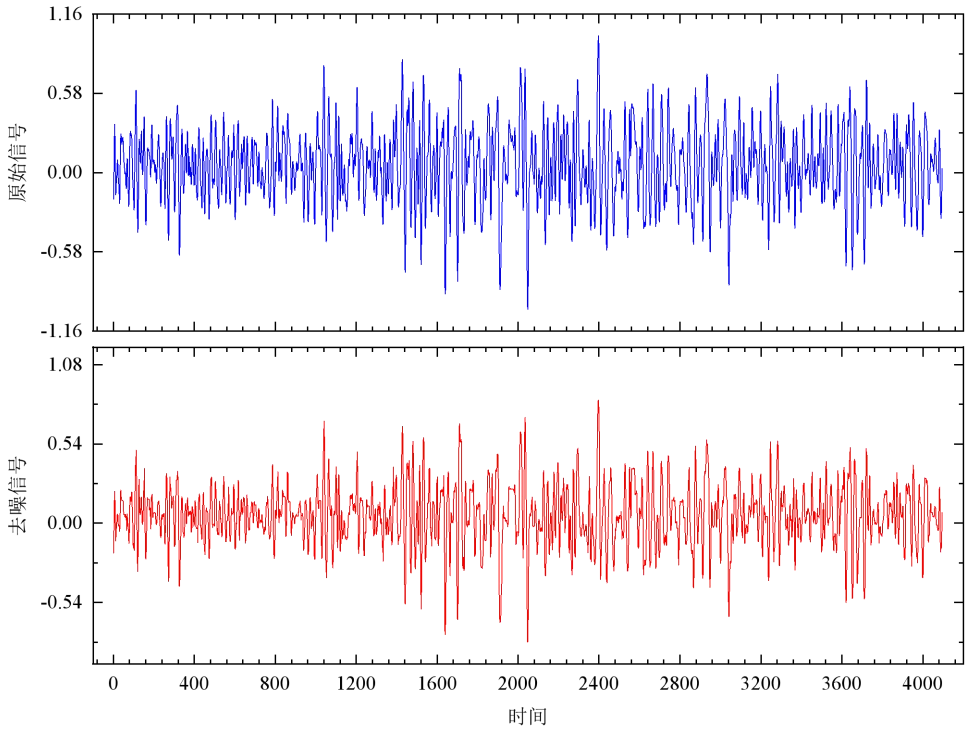

图 3 附件一的原始信号和去噪信号

分别采用了双树复小波变换和其他经典的小波变换对数据信号进行处理，处理过程中均采用最小极大方差。从图 4 中可以看出，经过双树复小波变换降噪比其他经典小波降噪去噪效果更明显，效果较好。

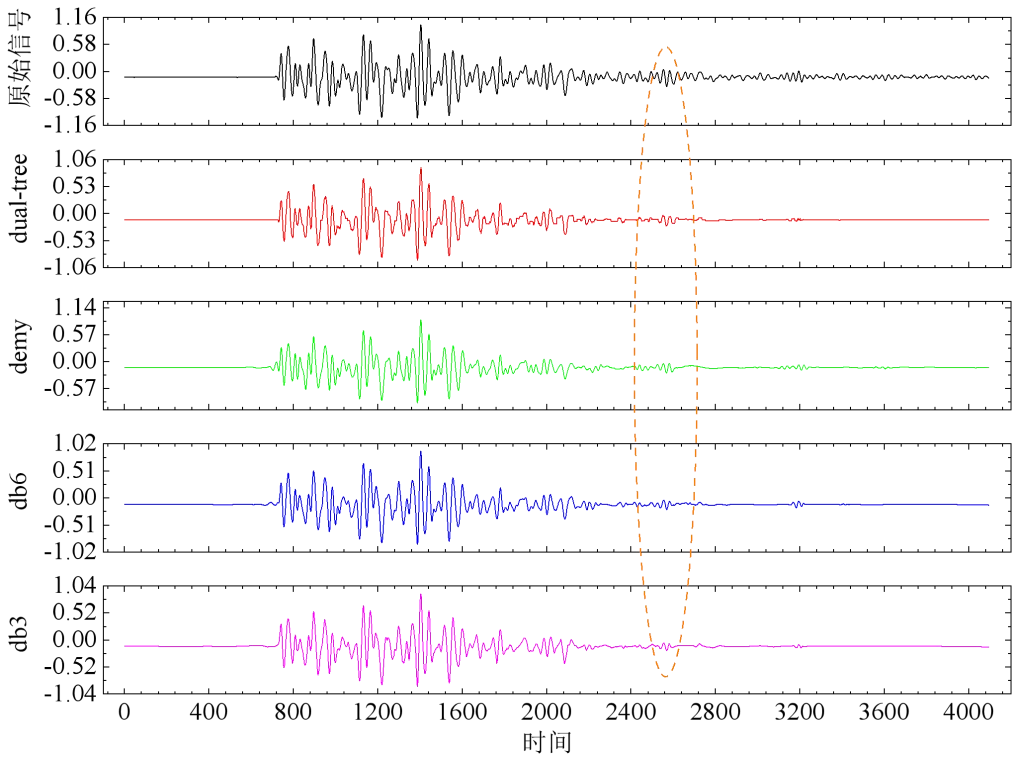

图 4 四种去噪模型去噪效果对比图

以双树复小波变换去噪模型，分别对附件一和附件二的数据进行了测试，评价对比结果如表 2、表 3 所示。附件一中，双树复小波变换去噪相比较与其他经典小波去噪 *MSE* 降低了约 25%，*SSE* 约 23%，*RMSE* 约 13%，信噪比提升约 0.9dB。附件二中，双树复小波变换去噪相比较与其他经典小波去噪 *MSE* 降低了约 7%，*SSE* 约 9%，*RMSE* 约 5%，信噪比提升约 0.3dB，可视化如图 5、图 6 所示，说明了双树复小波去噪的优越性。

表 2 附件一 信号去噪误差和信噪比结果表

降噪算法	MSE	SSE	RMSE	SNR	MAE	MAPE	R2
db3	0.00424	17.36120	0.06510	10.30626	0.05416	77.90321	0.93942
db6	0.00536	21.93919	0.07319	9.04408	0.06089	72.70190	0.92344
db9	0.00534	21.88298	0.07309	9.06102	0.05991	79.09253	0.92364
rbio1.1	0.00430	17.61322	0.06558	10.26273	0.05405	75.06073	0.93854
dmey	0.00454	18.59358	0.06738	10.03757	0.05458	81.48904	0.93512
dual-tree	0.00401	16.43317	0.06334	10.73360	0.05261	66.92818	0.94266

表 3 附件二 信号去噪误差和信噪比结果表

降噪算法	MSE	SSE	RMSE	SNR	MAE	MAPE	R2
db3	0.00255	10.45169	0.05051	9.58720	0.03666	516.48080	0.92619
db6	0.00260	10.66723	0.05103	9.46803	0.03711	137.29004	0.92466
db9	0.00284	11.64667	0.05332	8.88341	0.03843	836.69948	0.91775
rbio1.1	0.00263	10.75965	0.05125	9.37865	0.03735	190.94663	0.92401
dmey	0.00278	11.39215	0.05274	9.11754	0.03755	268.30347	0.91954
dual-tree	0.00255	10.44957	0.05051	9.61650	0.03694	834.35078	0.92620

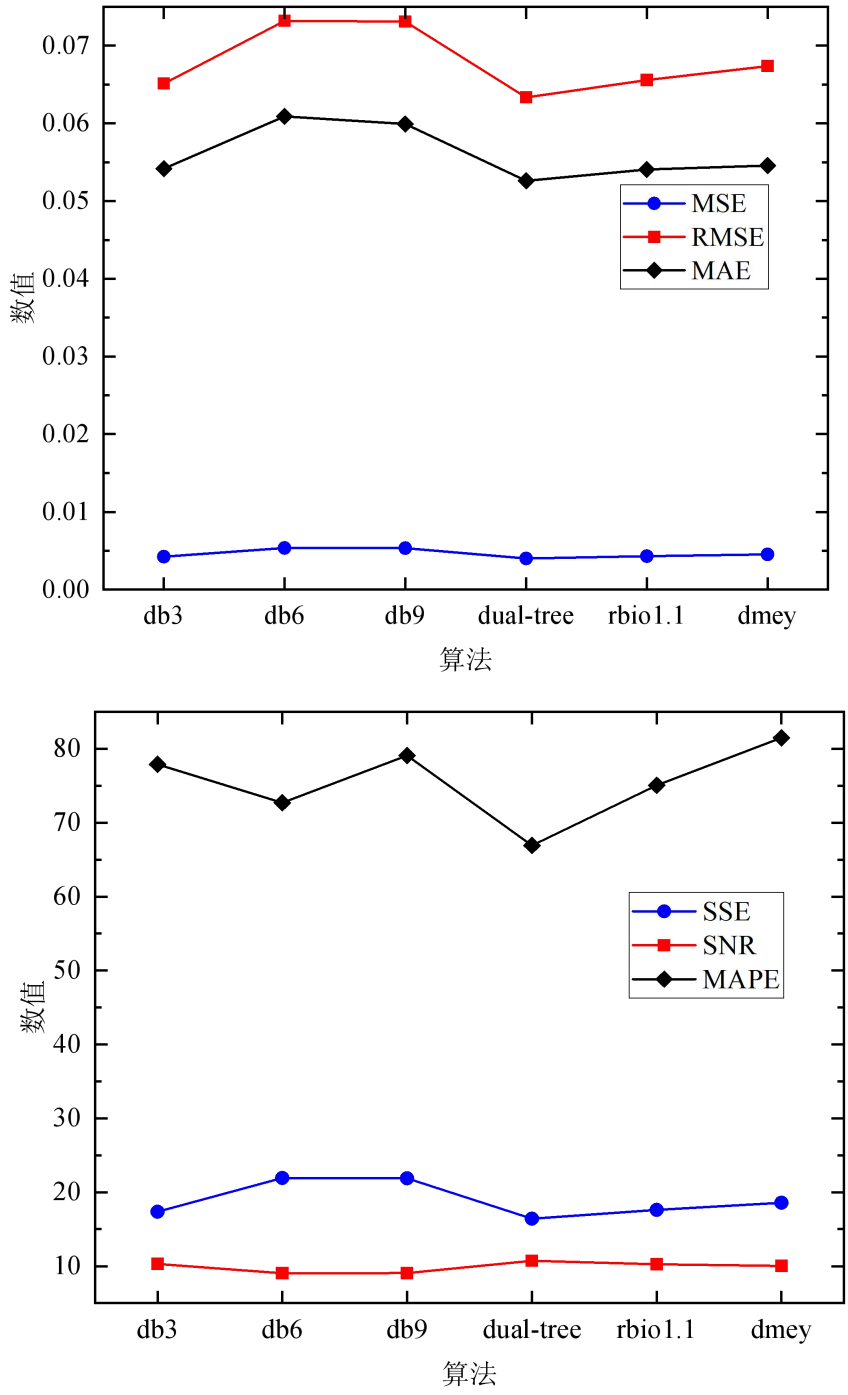

图 5 信号 4 不同算法去噪信号效果评价对比图

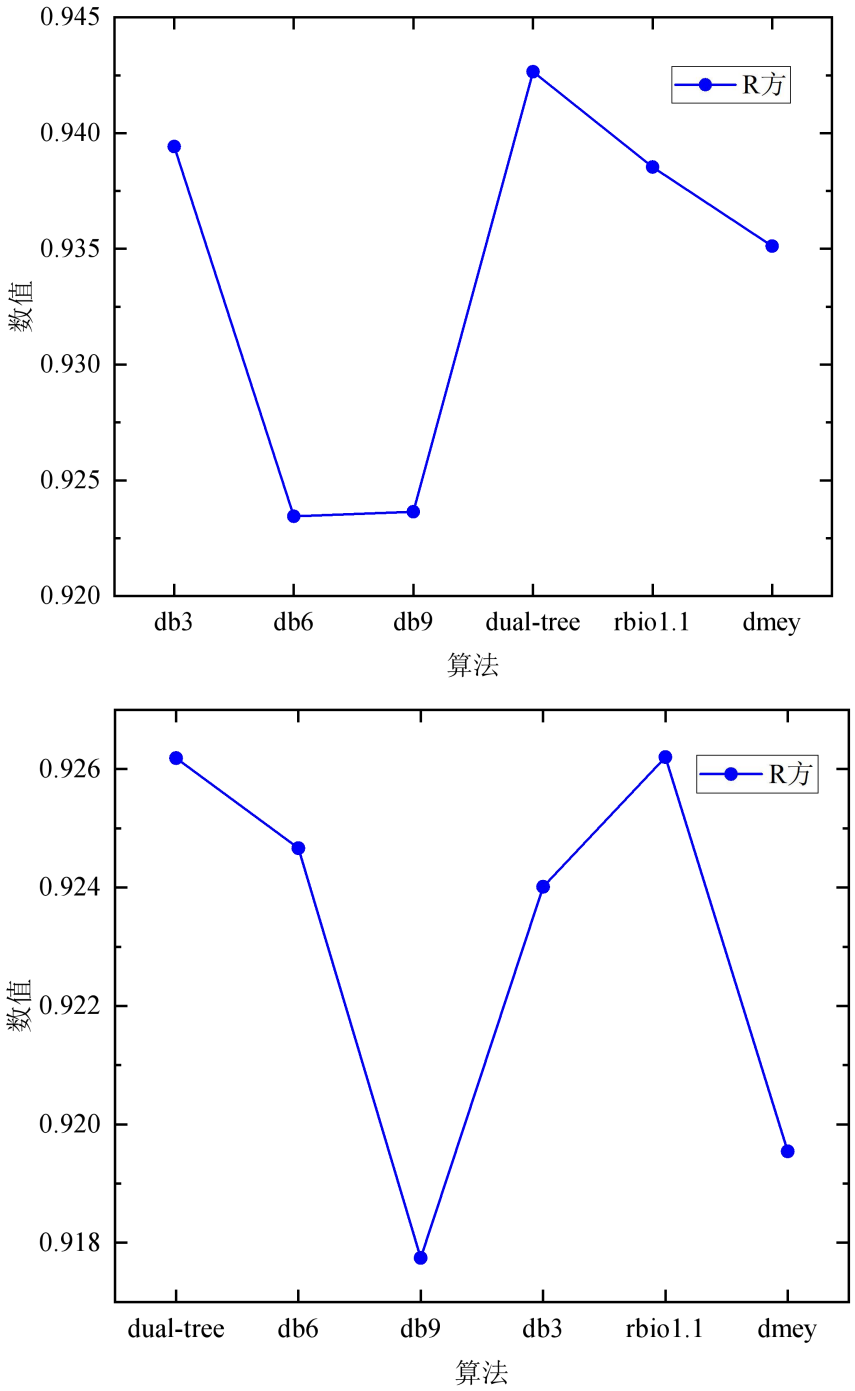

图 6 两种信号不同算法去噪信号效果 R 方对比图

六、 特征提取

6.1 特征提取方法

常用的特征提取方法分为三大类，包括时域特征提取，频域特征提取，以及时频域特征提取。

1) 时域特征提取通常包括的参数较多，比如有 RMS（有效值）、峰峰值、峭度、裕度、歪度、均值、均方根、脉冲因数、波形因数、波峰因数等等。
2) 频域特征提取主要包括频带能量提取和特征频率提取。
3) 时频域分析提取方法包括短时傅里叶变换和小波分析。时频域分析特别适用于分析非平稳信号，然后针对非平稳信号的特征提取可考虑时频域分析。

6.2 特征提取

本文将会根据输入的信号数据，输出二十八种特征，其中包含时域特征 15 类，频域特征 13 类。其中 15 个时域统计指标如下：

1) 均值：$\bar{x}=\frac{1}{N}\sum_{n=1}^{N}x(n)$
2) 绝对平均值：$|\bar{x}|=\frac{1}{N}\sum_{\mathrm{n}=1}^{N}|x_n|$
3) 方差：$\delta=\frac{1}{N}\sum_{n=1}^{N}x_n^2$
4) 标准差：$\sigma_x=\sqrt{\frac{1}{N-1}\sum_{n=1}^{N}[x(n)-\bar{x}]^2}$
5) 方根幅值：$x_{\mathrm{r}}=\left(\frac{1}{N}\sum_{n=1}^{N}\sqrt{|x(n)|}\right)^2$
6) 均方根值：$x_{\mathrm{ms}}=\sqrt{\frac{1}{N}\sum_{n=1}^{N}x^2(n)}$
7) 峰值：$x_{\mathrm{p}}=\max|x(n)|$
8) 最大值：$x_{\mathrm{rmax}}=\max(x_n)$
9) 最小值：$x_{\min}=\min(x_n)$
10) 波形指标：$W=\frac{x_{\mathrm{ms}}}{\bar{x}}$
11) 峰值指标：$I=\frac{x_{\mathrm{p}}}{\bar{x}}$
12) 脉冲指标：$L=\frac{x_{\mathrm{p}}}{x_{\mathrm{r}}}$
13) 裕度指标：$S=\frac{\sum_{n=1}^{N}[x(n)-\bar{x}]^3}{(N-1)\sigma_x^3}$
14) 偏斜度：$K=\frac{\sum_{n=1}^{N}[x(n)-\bar{x}]^3}{(N-1)\sigma_x^3}$
15) 峭度：$K=\frac{\sum_{n=1}^{N}[x(n)-\bar{x}]^4}{(N-1)\sigma_x^4}$

注意：$x(n)$为信号的时域序列，$n=1,2,\cdots,N$；N为样本点数。其中，第一类为时域统计特征，可直接通过振动信号的时间序列数据计算得出，如下列公式所示，其中：P_1、P_3—P_5反映了信号的时域幅值和能量特性；P_2、P_6—P_{11}反映了振动信号的时域分布特性。另外 13 个频域指标公式如下：

16) 频谱均值：$F_{16}=\frac{1}{K}\sum_{k=1}^{K}s(k)$

17) 频谱均方根值：$F_{17}=\sqrt{\frac{1}{K-1}\sum_{k=1}^{K}[s(k)-F_{16}]^2}$

18) $F_{18}=\frac{\sum_{k=1}^{K}[s(k)-F_{16}]^3}{(K-1)F_{17}^3}$

19) $F_{19}=\frac{\sum_{k=1}^{K}[s(k)-F_{16}]^4}{(K-1)F_{17}^4}$

20) 频率重心：$F_{20}=\frac{\sum_{i=1}^{K}f_i\cdot s(k)}{\sum_{k=1}^{K}s(k)}$

21) $F_{21}=\sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(f_k-F_{20})^2\cdot s(k)}$

22) 均方根频率：$F_{22}=\sqrt{\frac{\sum_{k=1}^{K}f_k^2\cdot s(k)}{\sum_{k=1}^{K}s(k)}}$

23) $F_{23}=\sqrt{\frac{\sum_{k=1}^{K}f_k^4\cdot s(k)}{\sum_{k=1}^{K}f_k^2\cdot s(k)}}$

24) $F_{24}=\frac{\sum_{k=1}^{K}f_k^2\cdot s(k)}{\sqrt{\sum_{k=1}^{K}s(k)\sum_{k=1}^{K}f_k^4\cdot s(k)}}$

25) $F_{25}=\frac{F_{21}}{F_{20}}$

26) $F_{26}=\frac{\sum_{k=1}^{K}(f_k-F_{20})^3\cdot s(k)}{(K-1)F_{21}^3}$

27) $F_{27}=\frac{\sum_{k=1}^{K}(f_k-F_{20})^4\cdot s(k)}{(K-1)F_{21}^4}$

28) 标准差频率：$F_{28}=\frac{\sum_{k=1}^{K}(f_k-F_{20})^{1/2}\cdot s(k)}{(K-1)F_{21}^{1/2}}$

$s(k)$是信号的频谱，$k=1,2,\cdots,K$；K是谱线数；f_k是第k条谱线的频率值。

其中F_{16}反映了信号的频域幅值和能量特征；$F_{17}-F_{19}$、F_{21}、$F_{25}-F_{27}$反映了信号的频域分布特征；F_{10}、$F_{22}-F_{23}$反映了振动信号的主要频率峰值。

七、 基于聚类方法的无监督学习故障检测分类模型

7.1 模型的建立

在本文中，我们选取了四个常用无监督学习的聚类模型进行实验，其中包括K-Means 模型、K-Means++模型、Mini Batch K-Means 模型、Birch 模型。

聚类模型中K值的选择是一个重要的实验过程，但是因为文章已经明确了数据类型分别为 A 和 B 两种类型，因此可直接确定K为 2 即可。

7.2 模型的求解

（1）K-Means++模型求解最优随机种子

由于 K-Means++初始化的时候，是随机初始化簇心，所以随机种子也会影响了聚类的效果。选择轮廓系数作为评价指标，根据样本 i 的簇内不相似度 a_i 和簇间不相似度 b_i，定义轮廓系数 s_i，计算方式如公式 16 所示。

$$s_i = \frac{b_i - a_i}{\max\{b_i, a_i\}} \tag{16}$$

轮廓系数范围在$[-1,1]$之间。该值越大，越合理,当s_i接近 1，则说明样本i聚类合理；当s_i接近-1，则说明样本i更应该分类到另外的簇；若s_i近似为 0，则说明样本i在两个簇的边界上。随机种子在 2000 到 2025 间对比了轮廓系数，如图 6 所示，当随机种子为 2016 时，轮廓系数最大。

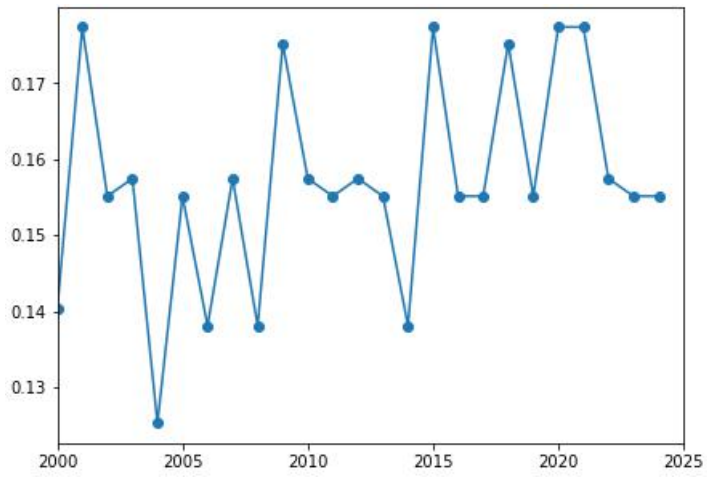

图 6 随机种子

（2）K-Means++聚类模型进行训练

将类型为 A 和 B 的数据进行 K-Means++聚类。K-Means++的算法过程如下：

第一步：从数据集中随机选取一个样本作为初始聚类中心c_1；

第二步：首先计算每个样本与当前已有聚类中心之间的最短距离，用$D(x)$表示；接收计算每个样本被选择下一个聚类中心的概率$\frac{D(x)^2}{\Sigma_{x\in X} D(x)^2}$。最后按照轮盘赌法选择下一个聚类中心；

第三步：重复第二步，直到选择出共K个聚类中心；

第四步：针对数据集中每一个样本x_i，计算它与K个聚类中心的距离并将其分到距离最小的聚类中心的类中；

第五步：针对每个类别c_i，重新计算它的聚类中心$c_i = \frac{1}{|c_i|}\sum_{x \in c_i} x$；

第六步：重复第五和第六步骤，直到聚类中心的位置不再发生变化。

K-Means++聚类效果三维可视化效果如图 7 所示。

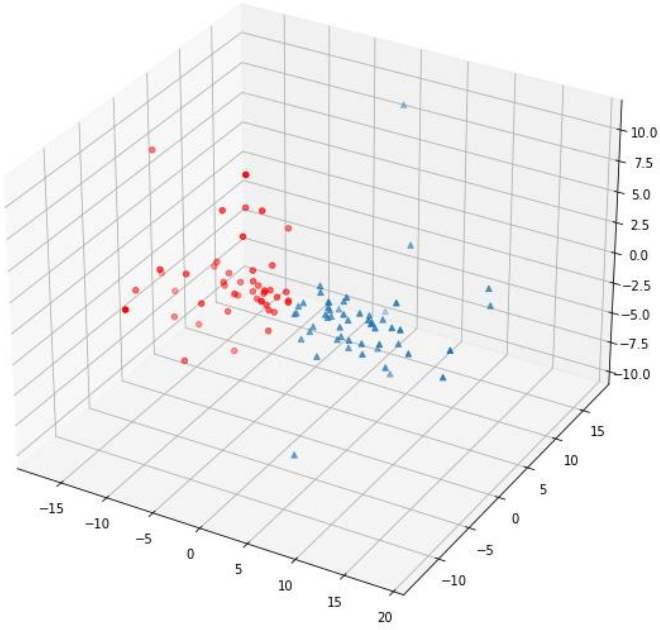

图 7 K-Means++聚类效果三维可视化图

（3）K-Means 聚类模型进行训练

K-Means 模型较为类似 K-Means++模型。虽然相比较而言，K-Means++的性能更优于 K-Means 模型。但是本文也对 K-Means 模型进行了实验，具体算法步骤如下：

1) 选择初始化的k个样本作为初始聚类中心$a = a_1, a_2, \cdots, a_k$；
2) 针对数据集中每个样本x_i计算它到k个聚类中心的距离并将其分到距离最小的聚类中心所对应的类中；
3) 针对每个类别a_j，重新计算它的聚类中心$a_j = \frac{1}{|c_i|}\sum_{x \in c_i} x$（即属于该类的所有样本的质心）；
4) 重复上面 2、3 两步操作，直到某个中止条件（迭代次数、最小误差变化等）。

其训练后效果图如图 8 所示：

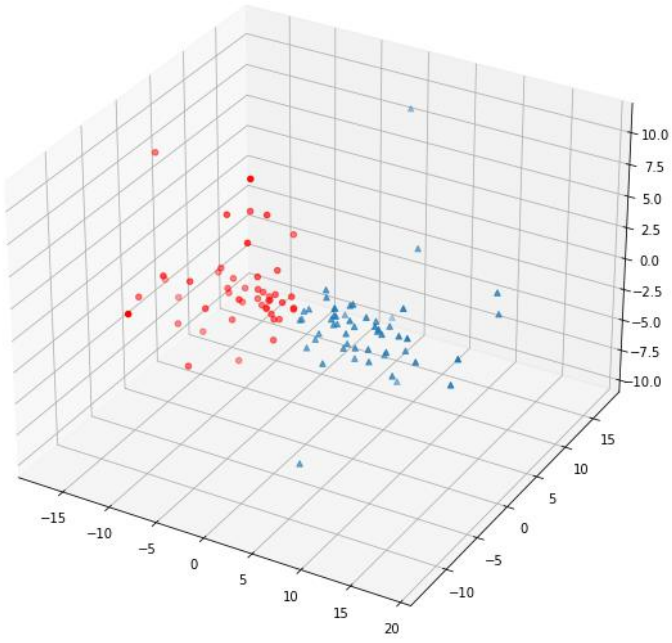

图 8 K-Means 聚类效果三维可视化图

（5）Mini Batch K-Means 聚类模型进行训练

MiniBatchK-Means 算法是 K-Means 算法的一种优化变种，采用小规模的数据子集来减少计算时间，同时试图优化目标函数；MiniBatchK-Means 算法可以减少 K-Means 算法的收敛时间，而且产生的结果只是略差与标准 K-Means 算法。

算法步骤如下：

1) 首先抽取部分训练集，使用 K-Means 算法构建出 K 个聚簇点的模型；
2) 继续抽取训练数据集中的部分数据集样本数据，并将其添加到模型中，分配给距离最近的聚簇中心点；
3) 更新聚簇的中心点值；
4) 循环迭代第二步和第三步操作，直到中心点稳定或者达到迭代次数，停止计算操作；

其训练后效果图如图 9 所示：

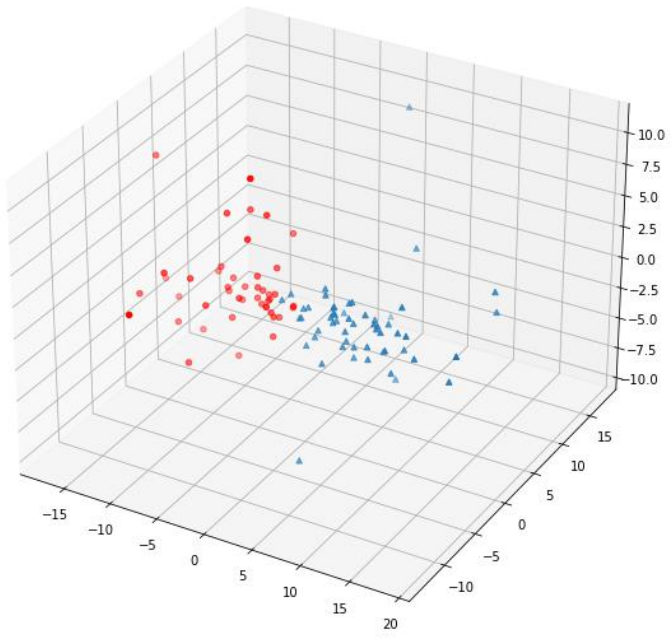

图 9 MiniBatchK-Means 聚类效果三维可视化图

（6）Birch 聚类模型进行训练

BIRCH 算法全称是 Balanced Iterative Reducing and Clustering Using Hierarchies。是属于树状结构的层次聚类算法的一种，其树状结构的构建是自上而下的，也就是说我们只需要扫描一遍数据，就可以得到树状结构了，因此该算法的运行速度很快。

算法步骤如下：

1) 扫描所有数据，建立初始化的 CF 树，把稠密数据分为簇，稀疏数据作为孤立点对待。
2) 补救由于输入顺序和页面大小带来的分裂，使用其他聚类算法对全部叶结点进行聚类。

可视化效果如图 10 所示：

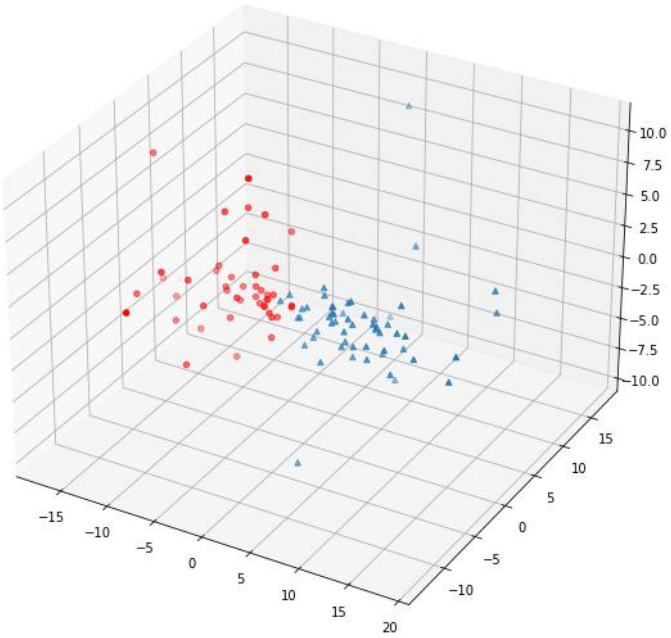

图 10 BIRCH 聚类效果三维可视化图

7.3 评价指标

（1）准确率

指在分类中，使用测试集对模型进行分类，分类正确的记录个数占总记录个数的比例，计算公式如下所示：其中，$n_{correct}$ 代表分类正确的记录个数，n_{tatal} 代表全部测试数据的个数。

$$Accuracy = \frac{n_{correct}}{n_{total}} \tag{17}$$

（2）召回率

召回率表示在所有的正类别样本中，被正确识别到的比例。

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

（3）轮廓系数

轮廓系数（Silhouette Coefficient），是聚类效果好坏的一种评价方式。最佳值为 1，最差值为-1，接近 0 的值表示重叠的群集[10]。负值通常表示样本已分配给错误的聚类，因为不同的聚类更为相似。其中对于其中的一个点 i 来说：计算簇内不相似度 $a(i)$，即 i 向量到同簇内其他点不相似程度的平均值可以体现凝聚度。而计算簇间不

相似度$b(i)$，即i向量到其他簇的平均不相似程度的最小值，可以体现分离度。具体公式可见公式（16）。

（4）F1 值

F1 分数（F1 Score），是统计学中用来衡量二分类模型精确度的一种指标。它同时兼顾了分类模型的精确率和召回率。F1 分数可以看作是模型精确率和召回率的一种加权平均，它的最大值是 1，最小值是 0。

$$F_1 = 2 \cdot \frac{precision}{precision + recall} \tag{19}$$

（5）Calinski-Harabaz 的 Score

Calinski-Harabaz 是评价聚类模型好坏的参考，常用于 K-Means 等聚类模型。对于聚类模型来说，我们希望聚类结果为相同类别之间的数据距离越近越好，而不同类别之间的数据距离越远越好；因此，对于k个聚类，Calinski-Harabaz 的分数s被定义为组间离散与组内离散的比率，该分值越大说明聚类效果越好。具体计算时，Calinski-Harbasz Score 是通过评估类之间方差和类内方差来计算得分。其计算公式如式（20）所示：

$$s = \frac{SS_B}{k-1} \Big/ \frac{SS_W}{N-k} \tag{20}$$

其中k代表类别数，N代表全部数据数目。SS_W是类间方差，SS_B是类内方差。

7.4 实验结果分析

表 4 说明了 TP、FN、FP、TN 等变量代表的实际意义。

表 4 变量说明

原故障类型	聚类后故障类型 A	聚类后故障类型 B
A	TP	FN
B	FP	TN

使用四种算法进行聚类后的结果如表 5—表 9 所示：

表 5 K-Means++结果

原故障类型	聚类后故障类型 A	聚类后故障类型 B
A	64	6
B	5	25

表 6 Birch 结果

原故障类型	聚类后故障类型 A	聚类后故障类型 B
A	61	9
B	7	23

表 7 K-Means 结果

原故障类型	聚类后故障类型 A	聚类后故障类型 B
A	61	9
B	9	21

表 8 MiniBatchK-Means 结果

原故障类型	聚类后故障类型 A	聚类后故障类型 B
A	59	11
B	7	23

表 9 不同聚类算法的评价指标对比

算法名称	准确率/%	召回率/%	耗时/ms	轮廓系数	F1-分数	CH 分数
K-Means	0.871	0.871	82.02	0.108	0.871	11.89
MiniBatchK-Means	0.894	0.843	16.60	0.104	0.868	8.0998
Birch	0.897	0.871	66.96	0.104	0.884	18.431
K-Means++	**0.928**	**0.914**	80.30	0.104	**0.921**	11.89

本部分首先根据任务的类型，选取了 Birch、K-Means、MiniBatchK-Means、K-Means++等几个常用于无监督分类的机器学习模型进行实验，并选择了精确率、召回率、耗时、轮廓系数、F1 分数、CH 分数等几个无监督模型评判指标。然后进行了随机种子实验，并确定无监督分类的种子设定系数，另外并对 K 值等训练系数进行了确定。之后进行了数据集的划分，7:3:1 的比例分别确定了训练集、验证集和测试集。训练完毕后在测试集上得到了上表所示的结果。可以看到，无论是准确率还是召回率，K-Means++相较于其他模型都有明显的优势。尽管 Birch、MiniBatchK-Means 等模型在耗时或者 CH 分数上占有优势，但是因为我们的最终目的是分类，更看重精确率等分类指标，因此本文最终选取了 K-Means++模型作为最终实验模型，经过多次试验后得到附表 3。

八、 基于有监督学习的故障检测二分类模型

8.1 模型的建立

针对本问题，采用机器学习二分类模型，分别有 XGBoost、CatBoost、LightGBM。特征的预处理依然采用问题一和问题二的处理过程。首先将附件一和附件二进行标准，附件一的数据设定标签编码为 0，附件二设定变迁编码为 1，再合并两者。随机划分 80%的数据作为训练集，20%的数据作为测试集。其次设定模型的参数，选择五折交叉验证训练方法，将提取的 28 维特征输入到分类模型中进行训练。并用测试集测试，选取准确率 Accuracy、召回率 Recall、耗时 T、F 值、AUC 分数、精准率 Precision 作为评价分类效果的好坏。

XGBoost 算法的原理如下[11]：

对于有n个样本、m个特征的数据集$\mathcal{D}=\{(\mathrm{x}_i,y_i)\}\ (|\mathcal{D}|=n,\mathrm{x}_i\in\mathbb{R}^m,y_i\in\mathbb{R})$，样本的预测值$\phi(\boldsymbol{x}_i)$是各棵决策树的结果$f_i(\boldsymbol{x}_i)$之和，假设生成了$K$棵决策树，则样本预测值如公式（21）所示：

$$\phi(\boldsymbol{x}_i)=\sum_{i=1}^{K}f_i(\boldsymbol{x}_i) \tag{21}$$

下文介绍决策树的生成过程。算法的损失函数为：

$$\mathcal{L}(\phi)=\sum_i l\left(y_i,\hat{y}_i\right)+\sum_k\Omega\left(f_k\right) \tag{22}$$

$$\Omega(f_k)=\gamma T+\frac{1}{2}\lambda\|\mathcal{W}\|^2 \tag{23}$$

式（22）中，$l(y_i,\hat{y}_i)$是样本损失，$\Omega(f_k)$是正则化函数，惩罚系数γ降低了树的叶节点数量T，$\mathcal{W}=(w_1,w_2,w_3,\cdots,w_T)$是全部叶节点权重构成的向量，$\lambda$系数则限制了叶节点权重的大小。

第t次选代生成的决策树f_t，降低了第$t-1$次迭代后的损失，如式（24）所示：

$$\mathcal{L}^{(t)}=\sum_{i=1}^{n} l\left(y_i,\hat{y}_i^{(t-1)}+f_t(\boldsymbol{x}_i)\right)+\sum_{i=1}^{t-1}\Omega\left(f_i\right)+\Omega(f_t) \tag{24}$$

对损失函数进行二阶泰勒展开，如式（25）、式（26）所示：

$$\mathcal{L}^{(t)}\simeq\sum_{i=1}^{n}\left[l\left(y_i,\hat{y}_i^{(t-1)}\right)+g_i f_t(\boldsymbol{x}_i)+\frac{1}{2}h_i f_t^2(\boldsymbol{x}_i)\right]+\sum_{i=1}^{t-1}\Omega\left(f_i\right)+\Omega(f_t) \tag{25}$$

$$g_i=\partial_{\hat{y}_t^{(t-1)}} l\left(y_i,\hat{y}_i^{(t-1)}\right),h_i=\partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i,\hat{y}_i^{(t-1)}\right) \tag{26}$$

$\text{leaf}(\boldsymbol{x}_i)=j$ 表示样本 x_i，被决策树 f_t，分类至第 j 个叶节点，决策树的结果 $f_i(\boldsymbol{x}_i)$ 为该叶节点的权重 w_j。去掉常量 $l\left(y_i,\hat{y}_i^{(t-1)}\right)$ 和 $\sum_{i=1}^{t-1}\Omega\left(f_i\right)$ 后，第 t 次选代生成的决策树把样本分类至 T 个叶节点后的损失为：

$$\mathcal{L}^{(t)}=\sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)w_j+\frac{1}{2}\left(\sum_{i\in I_j} h_i+\lambda\right)w_j^2\right]+\gamma T \tag{27}$$

式（27）中，$I_j=\{i|\text{leaf}(\boldsymbol{x}_i)=j\}$ 表示第 j 个叶节点中样本的索引。将叶节点 j 的损失关于 w_j，求偏导，求得使损失最小的叶节点权重 w_j^*：

$$w_j^*=-\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i+\lambda} \tag{28}$$

式（28）被称作叶节点权重计算公式。将 w_j^*，代入叶节点损失函数，得到叶节点 j 的损失为：

$$\mathcal{L}_{\text{befor}}=-\frac{1}{2}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i+\lambda}+\gamma \tag{29}$$

将式（29）作为结点的结构损失，将结点中的样本依第 i 个特征的值 σ 分为两个节点，即左节点 $S_L=\{\boldsymbol{x}_i\,|\,\sigma(\boldsymbol{x}_i)\le\theta\}$ 和右节点 $S_R=\{\boldsymbol{x}_i\,|\,\sigma(\boldsymbol{x}_i)>\theta\}$，$\theta$ 值称为分裂点。样本索引分别为 I_L 和 I_R，分裂后的结构损失为：

$$\mathcal{L}_{\text{affer}}=-\frac{1}{2}\left(\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda}+\frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda}\right)+2\gamma \tag{30}$$

故叶结点 j 以第 j 个特征的 θ 为分裂点获得的结构增益为：

$$\begin{aligned}\mathcal{L}_{\text{split}}&=\mathcal{L}_{\text{before}}-\mathcal{L}_{\text{after}}\\&=\frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda}+\frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda}-\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i+\lambda}\right]-\gamma\end{aligned} \tag{31}$$

式（31）称为分裂增益公式[12]，用于衡量一个分裂结点的好坏。分裂系数γ用于减小结构的复杂度，防止决策树过拟合。XGBoost 算法根据分裂增益公式在各个特征中寻找分裂点进行分裂，决策树构建完成后计算叶节点权重以更新样本预测值，下一次迭代则以新的样本预测值构建一棵决策树，最终以多颗决策树的预测结果之和作为最终的预测值。

8.2 模型的求解

（1）LGB

XGBoost 使用决策树对一个变量进行拆分，并在该变量上探索不同额切割点（按级别划分的树生长策略），而 LightGBM 则专注于按叶子节点进行拆分，以便获得更好的拟合（这是按叶划分的树生长策略）[13]。这使得 LightGBM 能够快速获得很好的数据拟合，并生成能够替代 XGBoost 的解决方案。从算法上讲，XGBoost 将决策树所进行的分割结构作为一个图来计算，使用广度优先搜索（BFS），而 LightGBM 使用的是深度优先搜索（DFS）。

（2）XGB

XGBoost 采用系数感知算法，利用系数矩阵，节省内存和节省计算时间[14]。采用近似树学习，这类学习方式能得到近似的结果，但是比完成的分支切割要节省很多时间。该模型能在一台机器上进行并行计算，在多台机器上进行类似的分布式计算。并且利用名为核外计算的优化方法，解决在磁盘读取时间过程的问题。将数据集分成多个块存放在磁盘中，使用一个独立的线程专门从磁盘读取数据并加载到内存中，这样一来，从磁盘读取数据和在内存中完成数据计算就能并行运行。XGBoost 还可以有效地处理缺失值，训练时对缺失值自动学习切分方向。基本思路就是在每次的切分中，让缺失值别被切分到决策树的左节点和右节点，然后通过计算增益得分选择增益大的切分方向进行分裂，最后针对每个特征的缺失值，都会学习到一个最优的默认切分方向。

（3）CatBoost

CatBoost 用来对类别特征进行编码的方法并不是新方法，是均值编码[15]。该方法已经成为一种特征工程方法，被广泛应用于各种数据科学竞赛中，如 Kaggle。均值编码，也称为似然编码、影响编码或者目标编码，可将标签转换为基于它们的数字，并与目标变量相关联。如果是回归问题，则给予级别典型的平均值转换标签。

在实验中模型参数设置如表 10 所示

表 10 模型参数设置

XGBoost 参数名称	参数取值	LightGBM 参数名称	参数取值	CatBoost 参数名称	参数取值
learning_rate	0.05	learning_rate	0.05	learning_rate	0.03
n_estimators	1000	boosting_type	'gbdt'	iterations	3000
objective	binary:logistic	objective	binary:logistic	objective	binary:logistic
max_depth	7	metric	'mae'	depth	7
feature_fraction	0.9	feature_fraction	0.9	l2_leaf_reg	4
subsample	0.8	bagging_fraction	0.8	loss_function	'MAE'
colsample_bytree	0.9	bagging_freq	5	eval_metri	'MAE'
alpha	0.3	num_leaves	31		
gamma	0	verbose	-1		

对于模型训练，本文采用k折率交叉验证，这种方法将可用数据划分为k个分区（k通常取 4 或 5），实例化k个相同的模型，将每个模型在$k-1$个分区上训练，并在剩下的一个分区上进行评估。在本文中k选取为 5。

8.3 评价指标

选择准确率*Accuracy*、召回率*Recall*、耗时T、F_1值、*AUC*分数、精准率*Precision*作为评价标准。

$$T = S_t - S_e \tag{32}$$

$$AUC = \frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1} - x_i) * (y_i + y_{i+1}) \tag{33}$$

式（32）中，S_t表示开始时间，S_e表示结束时间。x_i表示*FP*的数量，y_i表示*TP*的数量。准确率、召回率、F_1值、*AUC*分数、精准率越大，耗时越小，模型分类效果就越好。

8.4 实验结果分析

对于本文所采用的基于 XGBoost 的二分类模型，是采用正则化项防止过拟合，不仅使用到了一阶导数，还使用二阶导数，损失更精确，且 XGBoost 的并行是在特征粒度上的考虑了训练数据为稀疏值的情况，可以为缺失值或者指定的值指定分支的默认方向，这能大大提升算法的效率。还支持列抽样，不仅能降低过拟合，还能减少计算。对比了其他的集成学习二分类模型 LightGBM 和 CatBoost，准确率 Accuracy、召回率*Recall*、耗时T、F_1、*AUC*分数、精准率*Precision*如图 11 和表 11 所示。

表 11 分类模型对比表

模型选取	准确率%	召回率%	耗时 ms	F 值	AUC 分数	精准率%
XGBoost	100	100	175	1	1	100
CatBoost	100	100	3963	1	1	100
LightGBM	70	50	204	0.4117	0,5	35

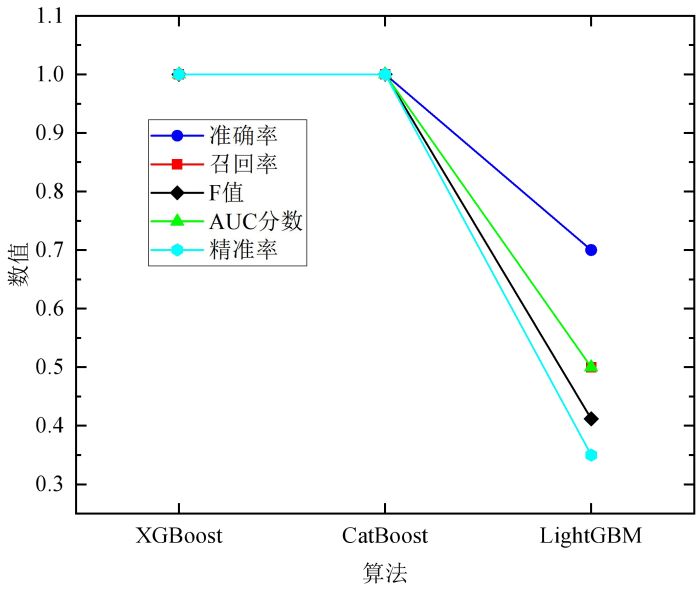

图 11 评价指标对比图

从图和表中可知，XGB 的准确率能达到 100%。相比较与 LightGBM 提升了约 30%。在相同的准确率下，CatBoost 耗时 3.9631 秒，XGB 耗费的时间为 0.1 秒，时间上提高了 97.4%。

九、 模型的检验

从表 12 可知，降噪处理后的数据集在聚类算法中，准确率、召回率、轮廓系数、CH 分数和 F1 值均提高了 1.5%左右，轮廓系数提高了 4.8%左右，模型训练耗费的时间较为接近，没有较大的差别。

表 12 聚类模型分析评价表

算法名称	Accuracy	Recall	Silhouette Coefficient	F1-score	Ch-score	TimesCost
K-Means	0.831	0.842	0.082	0.837	10.27	83
MiniBatch K-Means	0.868	0.843	0.092	0.644	10.28	167
Birch	0.881	0.843	0.087	0.660	17.11	69
K-Means++	0.913	0.900	0.099	0.906	18.83	73

从表 13 可知，降噪处理后的数据集，在分类算法中，准确率、召回率、轮廓系数、CH 分数、F1 值均模型训练耗费的时间较为接近，没有较大的差别。没有体现出差别来。原因可能是数据集的样本过少，模型难以体现出细微的差距。

表 13 分类模型评价对比表

分类模型/评价指标	准确率%	召回率%	耗时	F 值	AUC 分数	精准率
XGBoost	100.00	100.00	18.8	1.0000	1.0000	100.00
CatBoost	100.00	100.00	3837.6	1.0000	1.0000	100.00
LightGBM	70.00	50.00	306.5	0.4117	0.5000	35.00

十、 模型的评价与推广

针对问题一：与传统小波相比，双树复小波构造巧妙，并且具有平移不变性、抗频率混叠和可完全重构等优点。双树复小波变换和迭代奇异值分解相结合的分析方法比传统阈值降噪方法具有更好的降噪效果。信号经处理后，不仅能获得最大信噪比和最小均方值，还能够保证有用冲击成分不会被错误地滤除。除了能应用在实数信号上，还能应用在语音信号以及图像处理去噪中。

针对问题二：提取信号的特征过程，是发现重要特征，并将原始数据转化成更好的表达问题本质的特征的过程，使得将这些特征运用到模型中能提高对不可见数据的模型预测精度。

针对问题三：对于k个初始化的质心的位置选择对最后的聚类结果和运行时间都有很大的影响，因此需要选择合适的k个质心。如果仅仅是完全随机的选择，有可能导致算法收敛很慢。K-Means++算法就是对 K-Means 随机初始化质心的方法的优化。K-Means++原理比较简单，实现也是很容易，收敛速度快，在多种指定聚类数量的聚类算法中，效果较优，算法的可解释度比较强。能够应用在大部分的聚类任务中。如果各隐含类别的数据不平衡，比如各隐含类别的数据量严重失衡，或者各隐含类别的方差不同，则聚类效果不佳。缺点是该聚类方法对噪音和异常点比较的敏感，改进的方法又离群点检测的 LOF 算法，通过去除离群点后再聚类，可以减少离群点和孤立点对于聚类效果的影响。

针对问题四：XGBoost 简单易用。相对其他机器学习库，用户可以轻松使用 XGBoost 并获得相当不错的效果，可扩展高效，在处理大规模数据集时速度快效果好，对内存等硬件资源要求不高，鲁棒性强。相对于深度学习模型不需要精细调参便能取得接近的效果，XGBoost 内部实现提升树模型，可以自动处理缺失值。虽然然利用预排序和近似算法可以降低寻找最佳分裂点的计算量，但在节点分裂过程中仍需要遍历数据集，预排序过程的空间复杂度过高，不仅需要存储特征值，还需要存储特征对应样本的梯度统计值的索引，相当于消耗了两倍的内存。

结论

针对问题一：为了信号的平滑去噪，本文选择了双树复小波变换去噪方法，首先采用 Q 平移法构造双树复小波高低通滤波器，对原始信号进行 3 层双树复小波分解，得到高低频信号分量；然后采用最小极大方差软阈值方法对分解信号进行降噪处理；最后对降噪信号进行重构。并对比了基于 db3、db6、demy 小波基的小波变换，附件一中，双树复小波变换去噪相比较与其他经典小波去噪 降低了 MSE 约 25%，SSE 约 23%，RMSE 约 13%，信噪比提升约 0.9dB。附件二中，双树复小波变换去噪相比较与其他经典小波去噪 MSE 降低了约 7%，SSE 约 9%，RMSE 约 5%，信噪比 SNR 提升约 0.3dB。说明了双树复小波去噪的优越性。

针对问题二：本文直接提取信号的的时域特征以及采用快速傅里叶变换算法提取频域特征，共计 28 个。时域特征包括均值、标准差、方根幅值、均方根、峰值、峭度、峰值因子、裕度因子、波形因数、脉冲指数等。频域特征包括频谱均值、频谱均方根、均方根频率、标准差、特征频率、平均频率、均方频率、重心频率、频率方差。

针对问题三：利用问题一的去噪算法对数据信号进行预处理，再对数据集以问题二中的特征提取方法做特征工程，随机划分 80%的数据作为训练集，20%的数据作为测试集，最后采用 K-Means++聚类算法聚类为两类，用测试集价评价聚类的效果，通过轮廓系数的评价标准进行调参，得到了最佳的聚类效果。还对比了 K-Means、Mini Batch K-Means 和 Birch 三种聚类算法，分别为 87.1%、89.4%和 89.7%，K-Means++聚类效果最佳，准确率达到 92.8%、召回率 91.4%、耗时 91 毫秒、轮廓系数 0.115、F 值 0.921、CH 分数 10.450。

针对问题四：首先将附件一和附件二的利用问题一的去噪模型预处理后合并成一个数据集，设定两种不同的标签，并以问题二的提取特征的方式，对数据集进行构造特征工程。随机划分 80%的数据作为训练集，20% 的数据作为测试集，再采用有监督学习算法进行 XGBoost 进行二分类学习，并对比了 LightGBM 和 CatBoost 两种集成学习的分类模型。XGBoost 和 CatBoost 准确率均为 100%，LightGBM 准确率 70%，XGBoost 耗时 17.5 毫秒，CatBoost 耗时 3963 毫秒。

针对问题五：将数据信号分为两类，经过降噪和未经过降噪的数据，分别应用于问题三的分类模型和问题二的分类模型，降噪过后的信号，在聚类算法中准确率提高了约 1.5%、轮廓系数约 4.8%，说明了降噪提高了信号特征的表示能力。但在分类算法中未体现出区别。

参考文献

[1] 张鹏, 冯淼, 张涛然. 智能制造装备设计与故障诊断[M]. 北京: 机械工业出版社, 2021: 9-16.

[2] 贾娜.磁悬浮导系统信号处理中小波分析的应用研究[J]. 信息技术与信息化, 2020(09): 174-176.

[3] 刘昶, 胡皓然, 陈庆等. 一种新型远距离无线双向通信检测系统研究[J]. 电测与仪表, 2020, 57(15): 53-58.

[4] 王红尧,吴佳奇,李长恒等. 矿用钢丝绳损伤检测信号处理方法研究[J]. 工矿自动化, 2021, 47(02): 58-62.

[5] 田劼, 王洋洋, 郭红飞等. 基于漏磁检测的钢丝绳探伤原理与方法研究[J]. 煤炭工程, 2021, 53(09): 95-100.

[6] 田劼, 宋姗. 改进粒子群优化小波阈值的矿用钢丝绳损伤信号处理方法研究[J]. 煤炭工程, 2020,52(04): 103-107.

[7] Torbati N, Ayatollahi A, Sadeghipour P. Image-based gating of intravascular ultrasound sequences using the phase information of dual-tree complex wavelet transform coefficients[J]. IEEE transactions on medical imaging, 2019, 38(12): 2785-2795.

[8] 艾树峰. 基于双树复小波变换的轴承故障诊断研究[J]. 中国机械工程, 2011, 22(20): 2446-2447.

[9] 王勇, 王臻. 基于双树复小波的移动阴影检测和移除[J]. 网络安全技术与应用, 2020(04): 54-56.

[10] 宋阳,石鸿雁.基于MapReduce框架下的K-means聚类算法的改进[J].计算机与现代化,2019(08):28-32+43.

[11] 李京泰, 王晓丹. 基于代价敏感激活函数 XGBoost 的不平衡数据分类方法[J/OL]. 计算机科学, 2022: 01(34):1-17.

[12] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016: 785-794.

[13] 雷明. 机器学习原理算法与应用[M]. 北京: 清华大学出版社, 2019: 43-45.

[14] 天池平台. 阿里云天池大赛赛题解析[M]. 北京: 电子工业出版社, 2021: 273-274.

[15] 王贺, 刘鹏, 钱乾. 机器学习算法竞赛实战[M]. 北京: 人民邮电出版社, 2021: 69-70.

附录

附录 A 问题一代码

```
    ############################################################
    # 问题一
    ############################################################

    import numpy as np
    import pywt
    import pandas as pd
    import matplotlib.pylab as plt
    import warnings # Supress warnings
    warnings.filterwarnings('ignore')
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号

    import numpy as np
    import math
    from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
    # 输入 S 为纯信号,是一个 numpy 的 1D 张量
    # 输入 SN 为带噪信号，是一个 numpy 的 1D 张量
    # 输出 snr 为信噪比，单位为 dB，是一个 32 为的 float 数
    def SNR(SN, S):
        # 其中 S 是纯信号，SN 是带噪信号，snr 是信噪比
        Ps = sum((S-(np.mean(S)))**2) #signal power
        Pn = sum((S-SN)** 2)# noise power
        snr = 10*math.log((Ps/Pn), 10)
        return(snr)

    def print_metric(y_test, y_predict):
        mse = mean_squared_error(y_test, y_predict)

        mae = mean_absolute_error(y_test, y_predict)
        mape = np.mean(np.abs((y_predict - y_test) / y_test)) * 100
        r2 = r2_score(y_test, y_predict)
        snr = SNR(y_test, y_predict)
        rmse = np.sqrt(((y_predict - y_test) ** 2).mean())
        sse = np.sum((y_test - y_predict) ** 2)
        print('MSE:{} SSE: {} RMSE:{} SNR:{} MAE:{} MAPE:{} R2:{} '.format(mse,
sse, rmse, snr, mae,mape, r2))
        return [mse, sse, rmse, snr, mae, mape, r2]

    # 使用小波分析进行阈值去噪声,使用 pywt.threshold
    import matplotlib.pyplot as plt
    import pywt
    # 均方误差 from sklearn.metrics import mean_absolute_error #平方绝对误差 from
sklearn.metrics import r2_score#R square#调用
```

```
    metric_list = []
    for obj in ['db2', 'db3', 'db4', 'bior2.2', 'rbio1.5', 'dmey']:
        # Get data:
        input_data = np.array(pd.read_csv(r"../data/{}.txt".format(file),
sep='  ', header=None))
        if file>4:
            ecg = input_data
        else:
            ecg = input_data.T
        index = []
        data = []
        for i in range(ecg.shape[0]-1):
            X = float(i)
            Y = float(ecg[i])
            index.append(X)
            data.append(Y)

        # 创建小波对象并定义参数:
        w = pywt.Wavelet(obj)  # 选用 Daubechies8 小波
        maxlev = pywt.dwt_max_level(len(data), w.dec_len)
        # print("maximum level is " + str(maxlev))
        threshold = 0.1  # Threshold for filtering

        # 分解为小波分量，直至选定的水平:
        coeffs = pywt.wavedec(data,obj, level=maxlev)  # 将信号进行小波分解

        plt.figure()
        for i in range(1, len(coeffs)):
            coeffs[i] = pywt.threshold(coeffs[i], threshold*max(coeffs[i]))  #
将噪声滤波

        datarec = pywt.waverec(coeffs, obj)  # 将信号进行小波重构

        mintime = 0
        maxtime = mintime + len(data) + 1
        print('{}-----------------------------'.format(obj))
        tmplist = print_metric(ecg.reshape(-1,), datarec)
        metric_list.append(tmplist)
    # 双树复小波 MATLAB
    clear all
    close all
    filename = '4.txt';
    x=load(filename)
    setfig(1);
    [Yl,Yh] = dtwavexfm2(x,1,'near_sym_b','qshift_b');
    yy = zeros(size(x) .* [2 3]);
    yt1 = 1:size(x,1); yt2 = 1:size(x,2);
    for mlev = 1:5,
    mask = zeros(6,5);
    mask(:,mlev) = 1;
    z = dtwaveifm2(Yl*mask(1,5),Yh,'near_sym_b','qshift_b',mask);
```

```
yy(yt1,yt2) = z;
yt2 = yt2 + size(x,2)/2;
end
[Yl,Yh] = wavexfm2(x,4,'antonini');
yt1 = [1:size(x,1)] + size(x,1); yt2 = 1:size(x,2);
for mlev = 1:5,
mask = zeros(3,5);
mask(:,mlev) = 1;
z = waveifm2(Yl*mask(1,5),Yh,'antonini',mask);
figure;draw(z);drawnow
yy(yt1,yt2) = z;
yt2 = yt2 + size(x,2)/2;
end
hold
```

附录 B 问题二代码

```
    ############################################################
    # 问题二
    ############################################################
    import pandas as pd
    import warnings  # Supress warnings
    warnings.filterwarnings('ignore')
    import numpy as np
    data = []
    for i in range(100):
        filepath = "../data/{}.txt".format(i+1)
        if i<70:
            txt = pd.read_csv(filepath,sep='  ', header=None).T
        else:
            txt = pd.read_csv(filepath, sep='  ', header=None)
        data.append(list(txt[0]))
    x = np.array(data)
    df = pd.DataFrame(data)
    df.to_csv('../data/1-100.csv',index=0)

    import scipy.stats
    class Fea_Extra():
        def __init__(self, Fs=25600):
            self.Fs = Fs

        def Time_fea(self, signal_):
            """
            提取时域特征 11 类
            """
            N = len(signal_)
            y = signal_
            t_mean_1 = np.mean(y)                                    # 1_均值（平
均幅值）
            t_std_2 = np.std(y, ddof=1)                              # 2_标准差
            t_fgf_3 = ((np.mean(np.sqrt(np.abs(y))))**2              # 3_方根幅值
            t_rms_4 = np.sqrt((np.mean(y**2)))                       # 4_RMS 均
方根
            # 5_峰峰值  (参考周宏锑师姐 博士毕业论文)
            t_pp_5 = 0.5*(np.max(y)-np.min(y))
            #t_skew_6   = np.sum((t_mean_1)**3)/((N-1)*(t_std_3)**3)
            t_skew_6 = scipy.stats.skew(y)                           # 6_偏度
skewness
            #t_kur_7   = np.sum((y-t_mean_1)**4)/((N-1)*(t_std_3)**4)
            t_kur_7 = scipy.stats.kurtosis(
                y)                          # 7_峭度 Kurtosis
            # 8_峰值因子 Crest Factor
            t_cres_8 = np.max(np.abs(y))/t_rms_4
            # 9_裕度因子  Clearance Factor
            t_clear_9 = np.max(np.abs(y))/t_fgf_3
            t_shape_10 = (N * t_rms_4)/(np.sum(np.abs(y))
```

```
                                        )          # 10_波形因子 Shape fator
            t_imp_11 = (np.max(np.abs(y))) / \
                (np.mean(np.abs(y)))  # 11_脉冲指数 Impulse Fator
            t_fea = np.array([t_mean_1, t_std_2, t_fgf_3, t_rms_4, t_pp_5,
                              t_skew_6,   t_kur_7,  t_cres_8,  t_clear_9,
t_shape_10, t_imp_11])

            #print("t_fea:",t_fea.shape,'\n', t_fea)
            return t_fea

        def Fre_fea(self, signal_):
            """
            提取频域特征 13 类
            :param signal_:
            :return:
            """
            L = len(signal_)
            PL = abs(np.fft.fft(signal_ / L))[: int(L / 2)]
            PL[0] = 0
            f = np.fft.fftfreq(L, 1 / self.Fs)[: int(L / 2)]
            x = f
            y = PL
            K = len(y)

            f_12 = np.mean(y)
            f_13 = np.var(y)
            f_14 = (np.sum((y - f_12)**3))/(K * ((np.sqrt(f_13))**3))
            f_15 = (np.sum((y - f_12)**4))/(K * ((f_13)**2))
            f_16 = (np.sum(x * y))/(np.sum(y))
            f_17 = np.sqrt((np.mean(((x - f_16)**2)*(y))))
            f_18 = np.sqrt((np.sum((x**2)*y))/(np.sum(y)))
            f_19 = np.sqrt((np.sum((x**4)*y))/(np.sum((x**2)*y)))
            f_20 =
(np.sum((x**2)*y))/(np.sqrt((np.sum(y))*(np.sum((x**4)*y))))
            f_21 = f_17/f_16
            f_22 = (np.sum(((x - f_16)**3)*y))/(K * (f_17**3))
            f_23 = (np.sum(((x - f_16)**4)*y))/(K * (f_17**4))

            #f_24 = (np.sum((np.sqrt(x - f_16))*y))/(K * np.sqrt(f_17))    #
f_24 的根号下出现负号，无法计算先去掉
            #print("f_16:",f_16)
            #f_fea = np.array([f_12, f_13, f_14, f_15, f_16, f_17, f_18, f_19,
f_20, f_21, f_22, f_23, f_24])
            f_fea = np.array([f_12, f_13, f_14, f_15, f_16, f_17,
                              f_18, f_19, f_20, f_21, f_22, f_23])
            #print("f_fea:",f_fea.shape,'\n', f_fea)
            return f_fea

        def Both_Fea(self, signal_):
            """
            :return: 时域、频域特征 array
```

```
        """
        t_fea = self.Time_fea(signal_)
        f_fea = self.Fre_fea(signal_)
        fea = np.append(np.array(t_fea), np.array(f_fea))
        #print("fea:", fea.shape, '\n', fea)
        return fea
data = np.array(pd.read_csv('../data/1-100.csv'))
pro_data = []
fe = Fea_Extra()
for i in range(len(data)):
    tmp_f = fe.Both_Fea(data[i])
    pro_data.append(tmp_f)
clear_data = pd.DataFrame(np.array(pro_data))
clear_data.to_csv('../data/clear_data.csv',index=0)
```

附录 C 问题三代码

```
    ############################################################
    # 问题三
    ############################################################
    # 2 聚类
    import time
    import pandas as pd
    import numpy as np
    from sklearn.cluster import K-Means
    from sklearn.metrics import silhouette_score
    import matplotlib.pyplot as plt
    from sklearn.decomposition import PCA

    from sklearn.cluster import
K-Means,MiniBatchK-Means,AgglomerativeClustering,Birch
    import xlwt
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import recall_score
    from sklearn.metrics import silhouette_score
    from sklearn.metrics import f1_score
    from sklearn.metrics import calinski_harabasz_score

    ## 数据读取
    array_all = []
    for i in range(100):
        file_path = 'A/'+str(i+1)+'.txt'
        inFile = open(file_path, 'r')#以只读方式打开某 fileName 文件
        if i+1<71:
            for line in inFile:
                trainingSet = line.split('    ') #对于每一行，按','把数据分开，这
里是分成两部分
            x = []
            for j in trainingSet:
                if '' == j:
                    continue
                if '  ' in j:
                    for k in j.split('  '):
                        if '' == k:
                            continue
                        x.append(float(k))
                else:
                    x.append(float(j))
        else:
            x = []
            for line in inFile:
                x.append(float(line))
        array_all.append(x)

    Train_data = pd.DataFrame(array_all)
```

```
    # 构造 x, y
    # from sklearn.utils import shuffle
    # X_data = shuffle(X_data)

    X_data = Train_data
    X_data = X_data.fillna(-1)
    Y_data = []
    for i in range(70):
        Y_data.append(1)
    for i in range(30):
        Y_data.append(0)
    Y_data = pd.DataFrame(Y_data)
    Y_data.columns=['4096']
    X_data = X_data.join(Y_data)

    X_data.drop('4096',axis=1, inplace=True)

    correct = 0
    re = []
    for j in range(100):
        y_pred = []
        time_start=time.time()
        # estimator = K-Means(n_clusters=2,random_state=2030)  # 构造聚类器
        # estimator =
K-Means(n_clusters=2,init='K-Means++',random_state=2030)  # 构造聚类器
        # estimator = MiniBatchK-Means(n_clusters=2,random_state=2030)  # 构造
聚类器
        estimator = Birch(n_clusters=2)  # 构造聚类器

        estimator.fit(X_data)
        time_end=time.time()
        # centers = estimator.cluster_centers_ # 两组数据点的中心点
        # labels = estimator.labels_   # 每个数据点所属分组

        y = np.array(Y_data)
        y = y.tolist()
        for i in range(100):
            citydai = X_data.loc[[i]]
            citydaima = np.array(citydai)
            citydaima = citydaima.tolist()
            predict = estimator.predict(citydaima)
            y_pred.append(predict)

            if predict[0] == y[i]:
                correct += 1
        y_true = y

        # 准确率
        acc = accuracy_score(y_true, y_pred)
        print("acc",acc)
```

```
        # 召回率
        recall = recall_score(y_true, y_pred, average='macro')
        print("recall",recall)

        # 耗时
        print('time cost',time_end-time_start,'s')

        # 轮廓系数
        sil = silhouette_score(X_data, estimator.labels_, metric='euclidean')
        print("轮廓系数",sil)

        # F 值
        f1 = f1_score(y_true, y_pred, average='macro')
        print("F 值",f1)

        # CH 分数
        ch = calinski_harabasz_score(y_true, y_pred)
        print("ch 分数",ch)
        al =
[str(round(acc,4)*100)+'%',str(round(recall,4)*100)+'%',round((time_end-time
_start)*1000,2),round(sil,3),round(f1,2),round(ch,2)]
        re.append(al)

    workbook = xlwt.Workbook(encoding='utf-8')
    sheet = workbook.add_sheet('pvuv_sheet')

    # 写入标题
    datas = re
    # datas = pd.DataFrame(datas)
    columns = ["acc","recall","time","sil","f","ch"]
    for col,column in enumerate(columns):
        sheet.write(0, col, column)
    # 写入每一行
    for row, data in enumerate(datas):
        for col, column_data in enumerate(data):
            sheet.write(row+1, col, column_data)

    workbook.save('问题 3.xls')
    clf0 = K-Means(n_clusters=2, random_state=2016)
    # clf0 = K-Means(n_clusters=2, init='K-Means++',random_state=2016)
    # clf0 = AgglomerativeClustering(n_clusters=2)
    # clf0 = Birch(n_clusters=2)

    pred0 = clf0.fit_predict(X_data)

    # score0 = silhouette_score(data0, pred0)
    pca = PCA(n_components=3)  # 输出两维
    newData0 = pca.fit_transform(X_data)  # 载入 N 维

    x1, y1, z1 = [], [], []
    x2, y2, z2 = [], [], []
```

```
x3, y3, z3 = [], [], []
for index, value in enumerate(pred0):
    if value == 0:
        x1.append(newData0[index][0])
        y1.append(newData0[index][1])
        z1.append(newData0[index][2])
    elif value == 1:
        x2.append(newData0[index][0])
        y2.append(newData0[index][1])
        z2.append(newData0[index][2])
    elif value == 2:
        x3.append(newData0[index][0])
        y3.append(newData0[index][1])
        z3.append(newData0[index][2])
# plt.subplot(132)
plt.figure(figsize=(10, 10))

# #定义坐标轴
ax1 = plt.axes(projection='3d')
ax1.scatter3D(x1, y1,z1, marker='^')
ax1.scatter3D(x2, y2,z2, marker='o',c='r')
plt.savefig('数据分布三维.png',dpi=300)
plt.show()
```

附录 D 问题四代码

```
    ############################################################
    # 问题四
    ############################################################
    import pandas as pd

    data = pd.read_csv('../data/clear_data.csv')
    from sklearn.metrics import precision_score
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import f1_score
    import time
    from sklearn.metrics import recall_score
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import f1_score, precision_score, recall_score

    def print_precison_recall_f1(y_true, y_pred):
        # 准确率
        acc = accuracy_score(y_true, y_pred)
        # 召回率
        recall  = recall_score(y_true, y_pred, average='macro')
        # F 值
        f1 = f1_score(y_true, y_pred, average='macro')
        # CH 分数
        auc = roc_auc_score(y_true, y_pred)
        # 精准率
        p = precision_score(y_true, y_pred, average='macro')
        # print("ACC:{} Recall:{} f1:{} AUC:{} Precision: {} ".format(acc,
recall,f1,auc,p))
        return [acc, recall, f1, auc, p]
    from sklearn.model_selection import train_test_split
    from catboost import CatBoostRegressor
    import lightgbm as lgb
    import XGBoost as xgb
    import numpy as np
    import time
    X  = data
    y = 70*[0]+30*[1]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=2022)
    time_start = time.time()
    lgbparams = {'num_leaves': 60,  # 结果对最终效果影响较大，越大值越好，太大会
出现过拟合
                 'min_data_in_leaf': 30,
                 'objective': 'binary',  # 定义的目标函数
                 'max_depth': -1,
                 'learning_rate': 0.03,
                 "min_sum_hessian_in_leaf": 6,
                 "boosting": "gbdt",
                 "feature_fraction": 0.9,  # 提取的特征比率
                 "bagging_freq": 1,
```

```
                "bagging_fraction": 0.8,
                "bagging_seed": 11,
                "lambda_l1": 0.1,  # l1 正则
                # 'lambda_l2': 0.001,     #l2 正则
                "verbosity": -1,
                "nthread": -1,  # 线程数量，-1 表示全部线程，线程越多，运行的速度越快
                'metric': {'binary_logloss', 'auc'},  # 评价函数选择
                "random_state": 2019,  # 随机数种子，可以防止每次运行的结果不一致
                # 'device': 'gpu' ##如果安装的事 gpu 版本的 lightgbm,可以加快运算
                }
    train_set = lgb.Dataset(X_train, y_train)
    val_set = lgb.Dataset(X_test, y_test)

    lgbmodel = lgb.train(lgbparams, train_set, num_boost_round=3000,
                         valid_sets=(train_set, val_set),
                         early_stopping_rounds=500,
                         verbose_eval=False)

    pred = lgbmodel.predict(X_test, predict_disable_shape_check=True)
    time_end = time.time()
    print('time = {}s'.format(time_end-time_start))
    print_precison_recall_f1(y_test, np.around(pred))
    time_start = time.time()
    xgbparams =  {'booster': 'gbtree',
                          'objective': 'binary:logistic',
                          'eval_metric': 'auc',
                          'max_depth': 4,
                          'lambda': 10,
                          'subsample': 0.75,
                          'colsample_bytree': 0.75,
                          'min_child_weight': 2,
                          'eta': 0.025,
                          'seed': 0,
                          'nthread': 8,
                          'silent': 1}

    dtrain = xgb.DMatrix(X_train, label=y_train)
    dtest = xgb.DMatrix(X_test)
    watchlist = [(dtrain, 'train')]
    bst = xgb.train(xgbparams, dtrain, num_boost_round=5, evals=watchlist)
    pred = bst.predict(dtest)
    time_end = time.time()
    print('time = {}s'.format(time_end-time_start))
    print_precison_recall_f1(y_test, np.around(pred))
    time_start = time.time()
    catmodel = CatBoostRegressor(
        iterations=3000, learning_rate=0.03,
        depth=7,
        l2_leaf_reg=4,
        loss_function='MAE',
```

```
        eval_metric='MAE',
        random_seed=2021)
    catmodel2 = catmodel.fit(X_train, y_train, verbose=False)
    pred = catmodel2.predict(X_test)

    time_end = time.time()
    print('time = {}s'.format(time_end-time_start))
    print_precison_recall_f1(y_test, np.around(pred))
    df_list = []
    for i in range(100):
        time_start = time.time()
        xgbparams = {'booster': 'gbtree',
                     'objective': 'binary:logistic',
                     'eval_metric': 'auc',
                     'max_depth': 4,
                     'lambda': 10,
                     'subsample': 0.75,
                     'colsample_bytree': 0.75,
                     'min_child_weight': 2,
                     'eta': 0.025,
                     'seed': 0,
                     'nthread': 8,
                     'silent': 1}
        X = data
        Y = 70*[0]+30*[1]
        dtrain = xgb.DMatrix(X, label=Y)
        # dtest = xgb.DMatrix(X)
        watchlist = [(dtrain, 'train')]
        bst = xgb.train(xgbparams, dtrain, num_boost_round=5, evals=watchlist)
        pred = bst.predict(dtrain)
        time_end = time.time()
        print('time = {}s'.format(time_end-time_start))
        result = print_precison_recall_f1(Y, np.around(pred))
        result.append((time_end-time_start)*1000)
        df_list.append(result)
    df_result = pd.DataFrame(df_list, columns=['准确率%', '召回率%', 'F1 值',
'AUC', '精准率%', '耗时/ms'])  # [acc, recall, f1, auc, p]
    df_data = df_result[['准确率%', '召回率%','耗时/ms', 'F1 值', 'AUC', '精准
率%']]
    df_data
    df_data.to_excel('./submit/附表 4.xlsx',index=0)
```