第九届湖南省研究生数学建模竞赛承诺书

我们仔细阅读了湖南省高校研究生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）与队外的任何人（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他公开的资料（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。

我们完全清楚，在竞赛中必须合法合规地使用文献资料和软件工具，不能有任何侵犯知识产权的行为。否则我们将失去评奖资格，并可能受到严肃处理。

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，我们将受到严肃处理。

我们授权湖南省研究生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）。

所属学校和学院（请填写完整的全名）：国防科技大学研究生院

参赛队员（打印后签名）：1. 黄倩

2. 丁浩权

3. 刘旭

指导教师或指导教师组负责人（打印后签名）：

日期：2024 年 7 月 4 日

（请勿改动此页内容和格式。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。）

使用智能手机记录人体活动状态

摘要

近年来，智能手机已深深融入人们的日常生活，成为必备品。随着机械电子技术的发展，各类高精度传感器不断小型化，使得智能手机在计算能力和存储空间快速增强的同时，集成了越来越多的传感器，例如三轴加速度计、陀螺仪、磁力计、温度计和气压计等。本文利用智能手机的强大计算能力和感知能力，研究基于智能手机多传感器数据融合的人体活动识别（HAR）技术。

针对问题一：对于实现活动状态分类，关键在于分类模型设计与特征提取。考虑到数据一致性的影响，本文首先将传感器采集的原始数据转换为适合分类的特征向量，并基于此数据提取 48 个时域特征及 42 个频域特征，设计了多决策树的随机森林分类算法。进一步基于问题 2 的 10 名实验人员的活动数据，验证了随机森林分类器在识别和分类活动状态上具有较高的准确性。

针对问题二：采用主成分分析对高维特征数据进行降维，建立基于 Stacking 融合的判别模型，Stacking 分类融合层包括基层和元层，基层采用 K 折交叉检验方法训练不同的基分类器，其对人体活动数据特征集中的每个样本进行预测得到识别结果，元层通过训练基层的分类结果得出较优的人体活动分类模型，用于对人体活动状态进行识别与分类。同时，将 K 最邻近、随机森林、支持向量机以及 Stacking 融合算法的准确性进行了对比。结果表明，Stacking 融合算法整体分类效果最好，对所有类别的准确率都很高，最低也有 0.91，其融合了多种分类算法的优势，显著提高了分类准确率。

针对问题三：针对不同人员在同一活动状态下的传感器数据是否可进行人员画像问题，本文考虑到传感器本身测量误差导致微弱特征信号被噪声掩埋问题，首先采用时频图分析方法，实现自适应小波降噪，并通过图像形态学方法提取数据的时频图图像特征，再结合问题 2 中的 Stacking 融合的判别模型，建立多特征融合加权的画像判别模型，进一步分析了传感器数据与实验人员年龄、身高、体重之间的关系，实现对人员画像的准确判别。

最后，对提出的模型进行全面的评价，这一模型贴合实际，能合理解决提出的问题，具有实用性强、算法效率高等特点。该模型还可进行多传感器融合、模型迁移、数据实时处理及参数在线进化，并应用于其他领域。

关键词：多特征融合　随机森林　Stacking 融合算法　时频分析　判别模型

目录

摘要.....I
1 问题综述.....1
1.1 问题背景.....1
1.2 问题描述.....1
1.3 问题意义.....2
2 问题分析.....2
3 模型假设与符号说明.....3
3.1 模型基本假设.....3
3.2 符号说明.....3
4 模型思路.....4
4.1 数据预处理.....5
4.2 数据特征.....7
4.2.1 时域特征.....7
4.2.2 频域特征.....7
4.2.3 时频图特征.....8
4.3 分类算法.....10
4.3.1 基于多特征融合的 stracking 算法.....10
4.3.2 基于时频图特征与加权融合的人员画像判别算法.....11
5 模型的建立与求解.....11
5.1 问题一模型的建立与求解.....11
5.1.1 基于传感器数据融合的时域特征提取.....11
5.1.2 基于随机森林的分类模型建立.....14
5.1.3 人员活动状态分类求解.....16
5.1.4 问题结论.....19
5.2 问题二模型的建立与求解.....20
5.2.1 基于 PCA 的特征降维.....20
5.2.2 基于 Stacking 融合的判别模型建立.....21
5.2.3 人员活动状态判别求解.....22
5.2.4 问题结论.....25
5.3 问题三模型的建立与求解.....26
5.3.1 基于时频图的人员画像特征构建.....26
5.3.2 基于多特征融合加权的画像判别模型建立.....29
5.3.3 人员画像判别求解.....30
5.3.4 问题结论.....31
6 模型评价与推广.....32
6.1 模型的优点与创新.....32

6.2 模型的不足 32
6.3 模型的改进 32
6.4 模型的推广 32
参考文献 34
附　录 36

1 问题综述

1.1 问题背景

人体动作识别（Human Action Recognition，HAR）具有广泛的研究意义和应用价值，旨在识别并标记人的行为。人体的动作和活动状态可以通过多种方式展现，如红外图像、点云、音频、加速度、雷达或 Wi-Fi 信号等。通过对这些不同类型的信息进行编码，提取出有用的数据，每种数据类型在不同场景下都有其独特优势。随着人们对运动、健康、安全及娱乐的关注增加，人体活动识别在日常生活中的应用变得越来越广泛。例如，它在智能家居、物理治疗、老人护理、辅助生活和生活记录等领域起到了重要作用。

智能手机的计算能力不断提升，加上其普及性，为人体活动识别技术的研究提供了极大的推动力。智能手机不仅集成了多种传感器，可用于人体活动识别研究，还具备较高的存储容量和强大的计算能力。因此，智能手机成为了人体活动识别的理想感知和计算平台。例如，华为手机的“华为运动健康”软件可以根据用户的跑步、步行、骑车和爬山等活动状态计算其每日消耗的热量，这类运动健康软件依赖智能手机记录的日常活动状态数据来精确计算用户的热量消耗。

1.2 问题描述

智能手机测量人体活动状态主要依赖于内置的运动传感器——加速度计和陀螺仪。加速度计用于测量手机在 X、Y、Z 三个轴向上的线性加速度变化。当手机发生加速度变化时，加速度计会感应到并将数据传输给手机处理器进行存储或分析。陀螺仪则用于测量手机绕着 X、Y、Z 轴的旋转角速度。当携带手机的人转向时，陀螺仪会感知到转向的速度和方向，并将这些数据传输给处理器进行存储或处理。通过加速度计和陀螺仪的数据，智能手机处理器可以感知手机的姿态、角度和方向的变化，并利用相应的算法判断或跟踪用户的活动模式。

实验邀请了 10 余名参与者携带运动状态传感器进行活动，并收集他们的日常活动数据。参与者需要完成“向前走、向左走、向右走、步行上楼、步行下楼、向前跑、跳跃、坐下、站立、躺下、乘坐电梯上行、乘坐电梯下行”这 12 种活动，每种活动记录了 5 组实验数据，每组数据包含数秒的线加速度和角加速度信息。现需要利用这些数据建立模型，解决以下问题：

（1）分类实验人员的活动状态：根据附件 1 中每人 60 组的活动数据，对每位实验人员的活动状态进行分类，并将结果记录在表 1 中。

（2）模型比较和分类准确度分析：利用问题 1 中的分类模型对 10 名实验人员的数据进行分类。然后，与问题 2 中的判别模型结果进行比较，分析分类准确度。接着，使用判别模型对附件 3 中某实验人员的 30 次活动状态数据进行分类，并将结果填写在表 2 中。

（3）分析特征差异：比较实验人员在年龄、身高、体重等方面的数据差异，分析不同人员在相同活动状态下的表现是否存在差异。基于分析结果，创建人物画像，并对附件 5 中的数据进行判别。

1.3 问题意义

现有的人体活动识别方法主要使用惯性传感器和单一分类算法[1,2]。单一类型传感器限制了识别种类和准确性，而单一分类算法未能充分利用多种分类算法的互补特性。Funtes 等[3]将智能手机获取的原始加速度数据转化为统计特征，并使用 SVM 分类器对站立、坐下和行走等活动进行分类识别，实验表明系统的总体准确率约为 93%。Fahim 等[4]构建了基于智能手机加速度传感器的 HAR 模型，对步行和跑步等活动具有较高的识别能力。文献[5]通过获取用户的上下文活动数据，分析并识别用户行为。此外，研究人员也将先进的深度学习技术应用于 HAR 研究。例如，Ronao 等[6]针对智能手机传感器和人体活动的特性，利用深度神经网络实现了有效的 HAR 技术。

本文研究的主要目标是设计基于智能手机多传感器数据融合的 Stacking 集成算法，实现对人体活动的准确识别，并建立基于多特征融合加权的画像判别模型。通过实验、对比和分析，设计了一种多传感器融合的人体活动识别架构方案，提高识别系统的准确性与鲁棒性，帮助人们准确识别活动状态，维护身心健康。

2 问题分析

为了提升人体活动识别的精准性和实时性，使其更好地服务于人们的日常生活，本文提出了一种基于智能手机多传感器融合技术的人体活动识别方法。该方法能够识别 12 种日常活动，包括“向前走、向左走、向右走、步行上楼、步行下楼、向前跑、跳跃、坐下、站立、躺下、乘坐电梯上行、乘坐电梯下行”。

首先，获取智能手机传感器（加速度计和陀螺仪）在不同人体活动下的数据。接着，通过数据过滤、归一化和滑动窗口分割等步骤，对多模态感知数据进行预处理和特征提取。然后，使用 Stacking 方法融合多种分类器，实现对人体活动的分类识别。接下来，利用公开的 MobiAct 人体活动识别数据集对系统进行测试，并与其他算法的实验结果进行对比分析。最后，采用本文提出的多特征融合检测算法解决三个具体问题。

这种方法不仅提高了人体活动识别系统的准确性和鲁棒性，还确保了实时性，使其在实际应用中更为实用和高效。通过多传感器数据融合与先进的机器学习算法结合，本文的方法展示了在多样化和复杂环境中进行人体活动识别的潜力，推动了该领域的发展，为人们的健康和生活提供了有力支持。

本文的技术路线如下图所示：

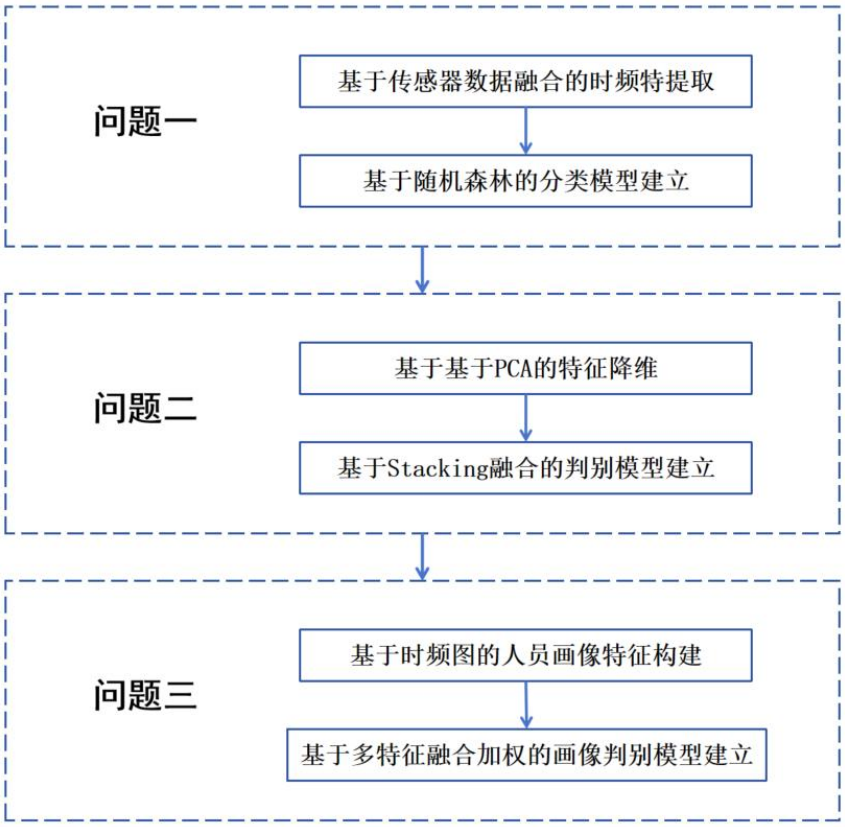

图 1 技术路线图

3 模型假设与符号说明

3.1 模型基本假设

（1）通过数据预处理和标准化后的活动数据可以提升模型的准确性。

（2）实验人员的活动数据可以根据 12 种活动状态进行分类。

（3）对活动数据提取的时域特征、频域特征以及时频图能够有效表示人员的活动状态信息。

（4）随机森林、支持向量机等机器学习模型能够有效地分类不同的人体活动状态。

（5）可以通过调参和交叉验证等方法选择最佳的模型和参数组合。

（6）训练数据和测试数据具有相似的分布，能够代表真实的使用场景。

（7）模型能够适应不同用户的个体差异，例如不同年龄、性别、体型等对人体活动状态的影响。

3.2 符号说明

本文定义了如下个使用次数较多的符号，其余符号在使用时注明。

表 1 符号说明

符号	含义

符号	含义
E	频谱能量
f_c	频谱中心
BW	频谱带宽
D	谱线密度
Q	谱线斜度
f_{peak}	谱线峰值
f_m	平均频率

4 模型思路

本文建立的模型主要用于处理运动传感器（加速度计和陀螺仪）采集的人体活动数据，通过提取显著特征对人的活动模式进行判断，在此基础上，实现基于多特征融合加权的人员画像判别。该模型在进行人体活动识别时主要分为数据预处理、数据特征提取和数据分类识别 3 个阶段。数据预处理包含滤波去噪、数据归一化以及数据分割，数据特征提取包括时域特征、频域特征以及时频图特征，并在此基础上采用 PCA 进行特征降维，数据分类识别包括基于 Stacking 融合的人体活动状态判别和基于多特征融合加权的人员画像判别。其中，Stacking 分类融合层包括基层和元层，基层采用 K 折交叉检验方法训练不同的基分类器，其对人体活动数据特征集中的每个样本进行预测得到识别结果。由于每个分类模型的训练精度在起初的时候上升较快，当训练精度达到一定程度，再对原始数据训练难以提高识别准确率，因此元层通过训练基层的分类结果得出较优的人体活动分类模型。

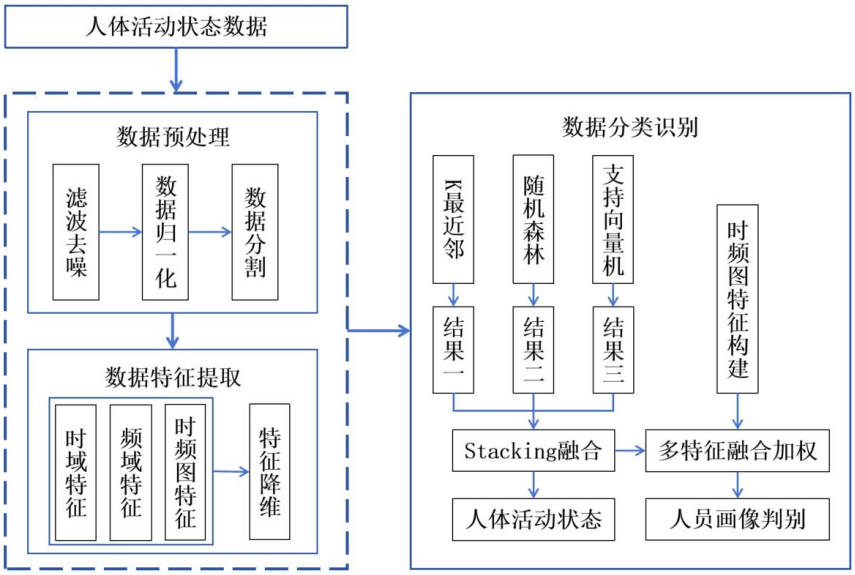

图 2 建模思路框图

4.1 数据预处理

数据预处理的目的是将传感器采集的原始数据转换为适合分类的特征向量。为了平滑数据并去除峰值信号的影响，我们采用中值滤波算法对多模态数据进行过滤和去噪。接着，对数据进行归一化处理，以提高训练速度和识别准确度。

（1）数据过滤

实际环境中采集的每种传感器数据都可能含有噪声，噪声既来自传感器本身，也来自实验人员，所以通过选择适当的滤波器对原始数据进行降噪，能较大程度的提高系统的识别效果。本文采用中值滤波算法对原始数据进行预处理，滤波前后加速度数据图形变化如图 3 和 4 所示，从图 3 明显的可以看出原始加速度数据的波形图存在较多的毛刺和抖动，图 4 是中值滤波处理后的数据变化图形，通过中值滤波算法对人体步行所获取的智能手机加速度传感器数据进行处理，消除了波峰，使得数据也更平滑，同时也保留传感器原始数据的数据特征。

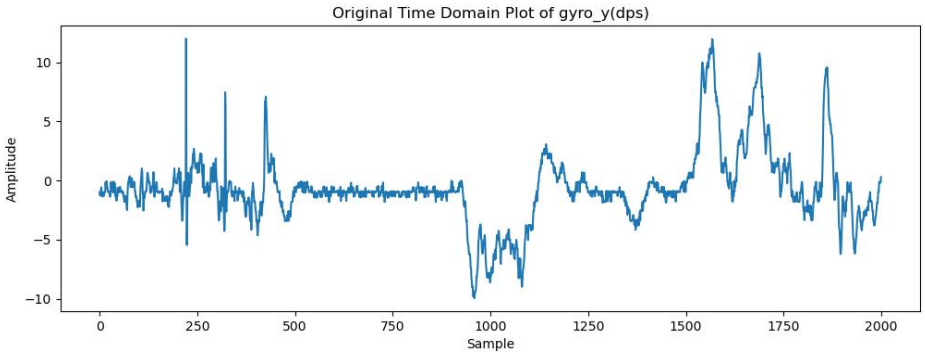

图 3 原始数据时序图

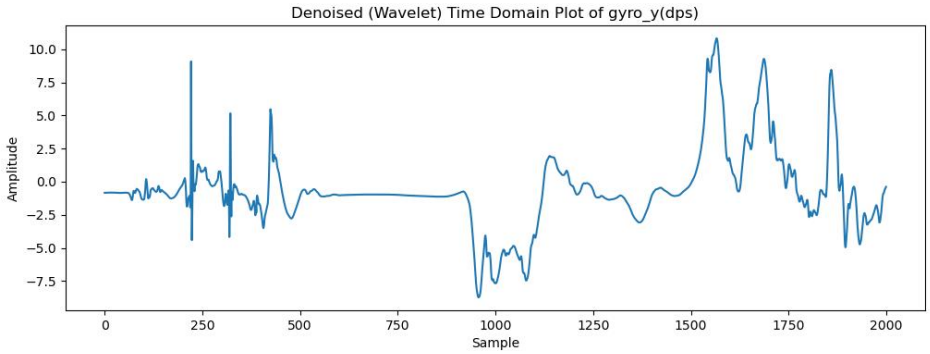

图 4 中值滤波后的时序图

（2）数据归一化

人体在进行一种活动时，由于获取的加速度和陀螺仪数据值的范围和单位不同，因此可能会降低识别的准确率。例如加速度单位为“米/秒²”，陀螺仪单位为“弧度/秒”，这样不同的量纲和单位不利于对数据进行分析。本文通过下列公式将加速度和陀螺仪传感器获得的原始数据归一化到[-1,1]之间。其中 x 代表传感器的原始数据，x_{mean} 代表原始数据的平均值，range 代表加速度或陀螺仪的量程范围，x_{scale} 代表归一化之后的数据。

$$x_{scale} = \frac{x - x_{mean}}{range}$$

如图 5 展示了人体走路动作产生的加速度数据归一化后的图形。通过图 5 可以总结归一化之后的数据只是在纵坐标上缩小了数值的大小，并没有改变数据的波形变化趋势。

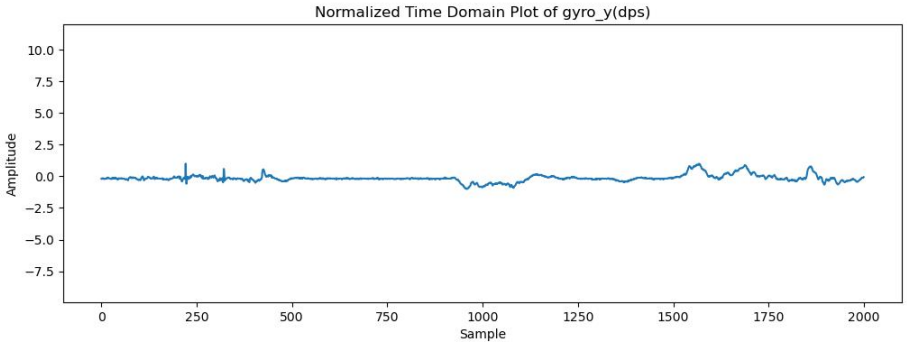

图 5 数据归一化后的时序图

（3）数据分割

为了减少人体活动分类识别的计算时间并准确识别活动详细信息，需要对原始传感器数据应用合适的数据分割方法，将数据分为一系列段。对传感器数据进行分割时，如果分割的窗口太小就不能同时包含足够的用于识别不同人体动作的信息，如果分割的窗口太大，系统识别人体活动数据时可能会出现较为严重的数据延迟和错误现象。本文采用滑动窗口分割技术，手机上的采集数据软件应用程序默认采样频率为 100Hz，滑动窗口大小设定为 2s，即用一个窗口的长度大小为 200 个样本点的滑动矩形窗分割原始加速度信号，相邻窗口按 50%进行数据的交叉分割和重叠。图 6 展示了对滤波后的走路加速度进行分割的结果，从图中可以明显的看出大小为 100 样本的窗口能够完全包含一段重复的波形，以便于本文接下来进行对数据特征的提取、选择和人体活动的识别。

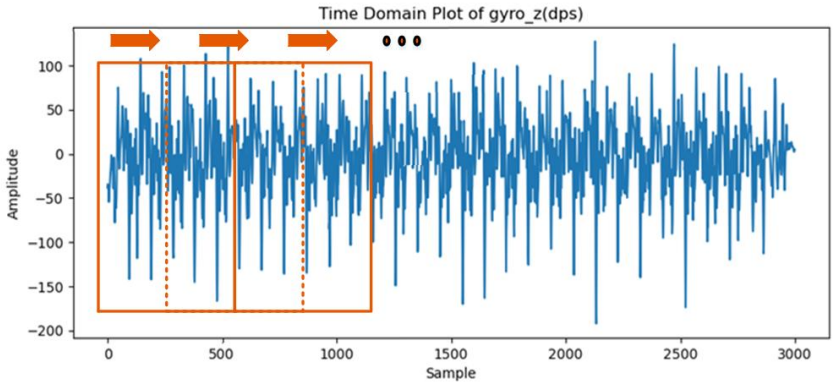

图 6 数据分割

4.2 数据特征

4.2.1 时域特征

时域特征通过提取数字信号或统计指标，展示信号随时间的变化。这些特征的主要优势在于计算时间较短，适用于在线和实时活动检测。从原始传感器数据中提取的时域特征向量通常具有集中趋势、可变性和信号形状分布的特点。

集中趋势度量特征描述信号接近中心值的情况，包括每个轴的平均值、最大值、最小值和中值。这些特征计算时间较短，已广泛应用于姿态识别、静态和动态活动检测以及能量消耗估计。可变性度量特征反映运动传感器信号在中心点之间的分布程度，变化越大，信号分布越广。包括标准偏差、变异、变异系数、四分位距、均方根、信号幅度、俯仰角和峰值幅度。这些特征计算成本低，能确定信号的稳定性和分布情况。形状信号分布特征有助于了解传感器信号的形状和分布，包括偏度和峰度。偏度测量信号的对称性，峰度反映数据分布的平坦度或峰值。这些特征广泛用于人类活动识别和健康监测。

综上所述，时域特征是传感器数据中独特且重要的特征，有助于准确识别人类活动。本文使用 8 种常见的时域特征提高分类性能，包括均值、方差、标准差、峰度、四分位距、均方根（RMS）、最大值和最小值。每个样本包含两个传感器（加速度计和陀螺仪）的数据，每个传感器有 x、y、z 三个维度的数据，因此每个样本可提取 48 个时域特征。

4.2.2 频域特征

基于频域特征提取是一种常见的信号处理方法，它主要依赖于信号在频率域上的特性进行分析和特征提取。频域分析是将信号从时域（时间域）转换到频域（频率域）的过程。这种分析方法通过傅里叶变换或其变种（如快速傅里叶变换 FFT）将信号表示为其在不同频率上的成分。频域特征是从信号的频谱表示中提取的数值特征，常见的频域特征包括频谱能量、频谱中心、频谱带宽、谱线密度、频谱斜度、频谱峰值以及平均频率等。

其中频谱能量表示为：

$$E = \sum_{f \in F} |X(f)|^2$$

频谱中心表示为：

$$f_c = \frac{\sum_{f \in F} f|X(f)|^2}{\sum_{f \in F} |X(f)|^2}$$

频谱带宽表示为：

$$BW = \sum_{f \in F} |X(f)|^2 \cdot (f - f_c)^2$$

谱线密度表示为：

$$D = \frac{\sum_{f \in F} |X(f)|^2}{\Delta f}$$

频谱斜度表示为：

$$Q = \frac{\sum_{f \in F} |X(f)|^2 \cdot (f - f_c)^n}{\sum_{f \in F} |X(f)|^2}$$

频谱峰值表示为：

$$f_{peak} = \arg\max_{f \in F} |X(f)|$$

平均频率表示为：

$$f_m = \frac{\sum_{f \in F} f|X(f)|^2}{\sum_{f \in F} |X(f)|^2}$$

基于频率的特征对于重复活动的分析在人体活动识别中是必需的，因此在每个分割的窗口上需要使用快速傅里叶变换（FFT）函数从原始信号数据中提取频域特征。从频域数据中，本文提取了频谱能量、频谱中心、频谱带宽、谱线密度、频谱斜度、频谱峰值以及平均频率 7 种特征向量。每个样本可提取 42，总 42 维特征向量。

4.2.3 时频图特征

时频图是在时间和频率上同时展示信号能量或功率的图像。它能够反映信号在不同时间段内的频率成分变化，提供了时间和频率信息的联合表达。基于时频图的特征提取是从时频表示中提取能够描述信号特性的数值特征。这些特征可以用来训练机器学习模型或进行后续的模式识别和分类任务。在处理人体活动数据时，时频图具有以下优势：

（1）捕捉时间和频率信息的联合变化。时频图能够同时展示信号在时间和频率上的变化，比传统的频域或时域分析更加全面和详细。这对于处理非稳态信号（如人体运动）和复杂环境中的数据非常有用。

（2）适应性强。时频图可以根据信号的动态性和复杂性进行调整和优化。例如，可以使用不同的窗口函数和分辨率来平衡时域和频域分辨率之间的权衡。

（3）提高特征的鉴别性和准确性。时频图特征能够提供更加丰富和具体的信号表达，因此有助于提高模型在分类、识别或预测任务中的准确性和鲁棒性。

（4）减少数据维度和复杂性。对于复杂的信号数据，时频图特征提取可以帮助减少数据的维度，同时保留关键的时间和频率信息，从而简化后续的数据处理和分析过程。

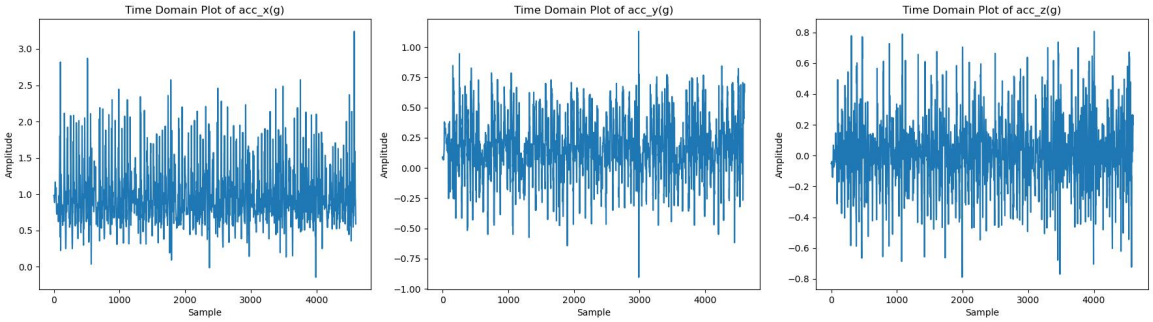

图 7 向前跑时加速度计测得的 x, y, z 方向数据时序图

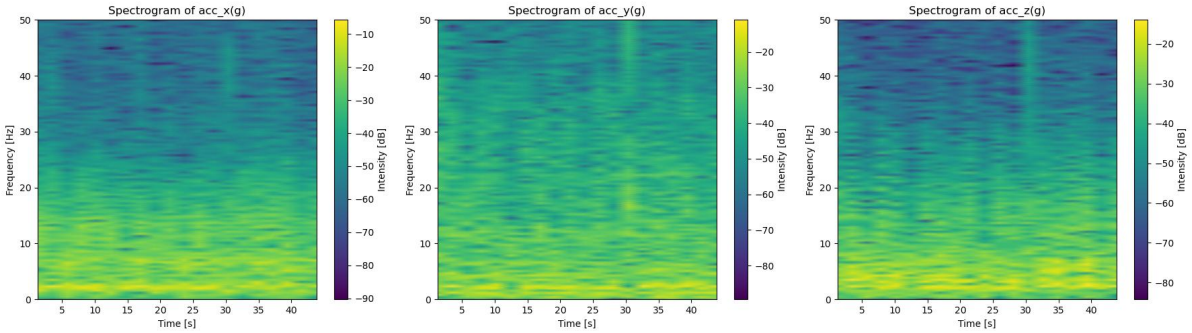

图 8 向前跑时加速度计测得的 x, y, z 方向数据时频图

图 7 展示了人员向前跑时加速度计测得的 x、y、z 三个方向的数据，图 8 为该数据对应的时频图。可以看出，时频图提供了人员向前跑过程中更丰富的信息。在 x 和 y 方向的数据中，主要能量集中在 0 到 20Hz 的频率范围内，且在整个时间段内频率成分和强度变化不大，主要集中在低频区域（0 到 10Hz）。随着时间的推移，能量的强度（以颜色表示）在低频区域较为集中，表明 x 方向上的运动主要集中在低频部分。在整个时间段内，z 方向的频率成分在 10 到 20 秒之间有明显变化，可能是跑步过程中某些特定动作导致的。低频区域（0 到 10Hz）同样有较强的能量集中，但相比 x 和 y 方向，z 方向在高频区域（10 到 20Hz）也有一定的能量分布。

4.3 分类算法

4.3.1 基于多特征融合的 stracking 算法

Stacking 是一种著名的集成学习方法，可以看作是一种特殊的组合策略。Stacking 模型本质上是一种分层结构，每一层可以包含多个模型，下一层利用上一层模型的结果进行学习。

当单个模型的训练精度达到瓶颈时，进一步提高精度变得困难。因此，本文不再单独训练原始数据，而是采用 Stacking 模型。具体来说，使用 K-最近邻算法、随机森林和支持向量机作为第一层模型，下一层模型直接拟合这些模型的输出，从而实现高效训练，提高算法的分类与判别能力。Stacking 通过多个基本模型对数据集进行预测，然后将预测结果加权平均，这些结果将作为新模型的训练集，进一步进行模型训练。基于多特征融合的 Stacking 算法如图所示：

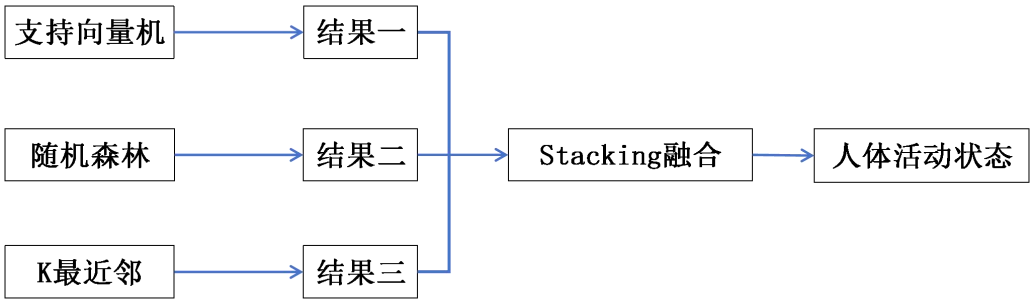

图 9 基于多特征融合的 Stacking 算法

（1）随机森林

随机森林是一个由许多决策树组成的集成模型。它的核心思路是，当训练数据被输入模型时，随机森林并不是用整个训练数据集建立一个大的决策树，而是采用不同的子集和特征属性建立多个小的决策树，然后将它们合并成一个更强大的模型。通过对多个决策树的结果进行组合，随机森林可以增强模型的效果。这种方法的主要优点是，通过集成多个模型，它可以有效地处理过拟合问题，提高模型的泛化能力。

（2）支持向量机

支持向量机（Support Vector Machine，SVM）是一种非常流行的监督学习方法，在许多领域中都有广泛的应用。该算法基于统计学习理论和线性代数的原理，具有较高的分类准确度和良好的鲁棒性。支持向量机不仅适用于线性可分问题，还可以通过引入核函数将样本映射到高维特征空间中，从而处理非线性可分问题。在高维特征空间中，支持向量机通过寻找一个最优的超平面来实现样本的有效分类。

（3）K 最近邻

K 最近邻（K-Nearest Neighbor，简称 KNN）算法是数据挖掘分类技术中最简单的方法之一。所谓的 K 最近邻指的是每个样本可以由其 K 个最近邻表示。

如果一个样本在特征空间中的 K 个最相邻的样本中大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。该方法在确定分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。KNN 方法在类别决策时，只与极少量的相邻样本有关。由于 KNN 算法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来判断所属类别的，因此对于类域的交叉，或重合较多的样本集来说，KNN 方法较其他方法更加适合。

4.3.2 基于时频图特征与加权融合的人员画像判别算法

实验中设备采样率为 100Hz。加速度计测量加速度变化范围为±6g，陀螺仪测量角速度变化范围为±500dps(degree per second)。一方面由于传感器本身测量必然存在误差且传统时域、频域方法难以消除，另一方面由实验人员年龄、身高、体重等差异带来的特征区别微小，所以会导致 12 组活动状态中人员画像的微小特征被噪声掩盖。

为解决分类识别研究中被噪声掩埋而很难考虑到的人员画像微小特征，本文提出基于时频图形态学提取特征与活动判别算法加权融合的人员画像判别方法。利用时频变换的分析方法，将一维信号处理成彩色时频图，实现一定程度上的信噪分离，并在此基础上引入图像形态学处理技术，将二维时频图进行二值化处理，建立基于轮廓特征的分类识别方法，并结合 Stacking 模型进行加权融合，实现人员画像的高准确率判别。其主要流程如下图所示：

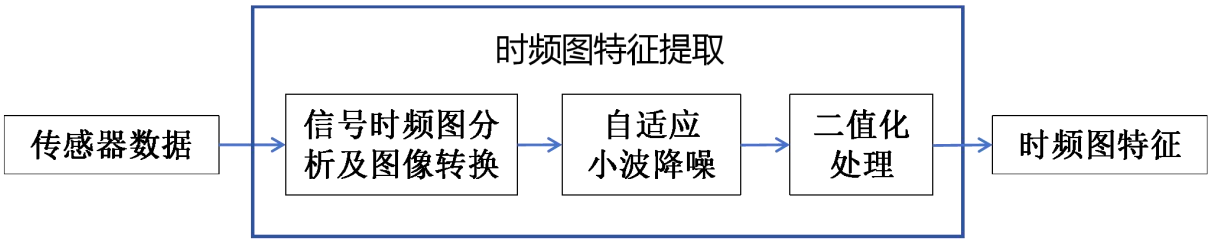

图 10 基于时频图特征与加权融合的人员画像判别算法

5 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 基于传感器数据融合的时域特征提取

采用基于加速度计和陀螺仪数据融合的方式对 12 种人体日常活动（向前走，向左走，向右走、步行上楼、步行下楼、向前跑、跳跃、坐下、站立、躺下、乘坐电梯向上移动、乘坐电梯向下移动）数据进行了对比和分析。下面分别为 12 类日常活动的加速度、角速度数据变化折线图。图 11-图 22 分别是 12 种活动状态时加速度和陀螺仪数据的图形显示。可以发现，行走、慢跑、上楼和下楼等行为的三轴加速度、角速度虽然表现出周期性变化规律，但变化规律却不尽相同，走路和慢跑在加速度数据的对比上较为类似，但是对比角速度变化趋势却明显不同。加速计测量的加速度力可动态感应运动和

振动并确保动态检测运动模式。陀螺仪传感器可测量角速度和方向，从而为检测相似模式（走路、慢跑）的活动和强位移活动提供补充信息。

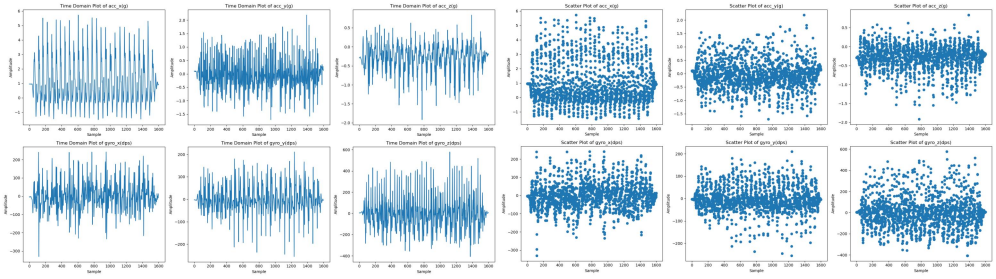

图 11 “向前走”时域图与特征提取

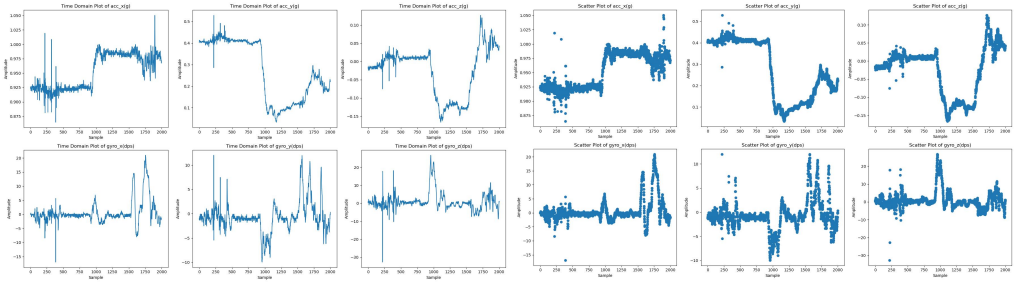

图 12 “向左走”时域图与特征提取

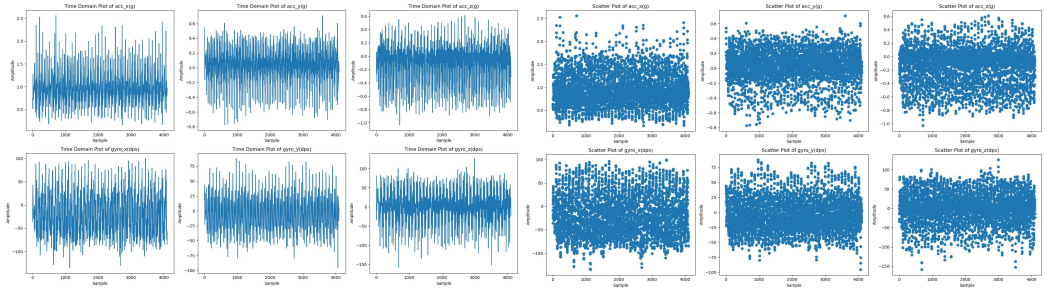

图 13 “向右走”时域图与特征提取

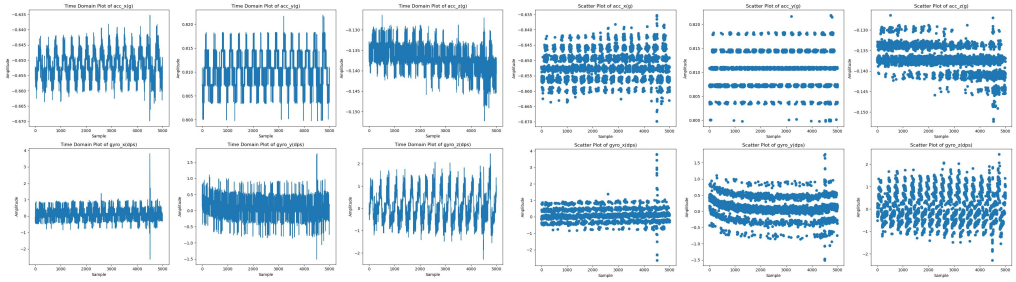

图 14 “步行上楼”时域图与特征提取

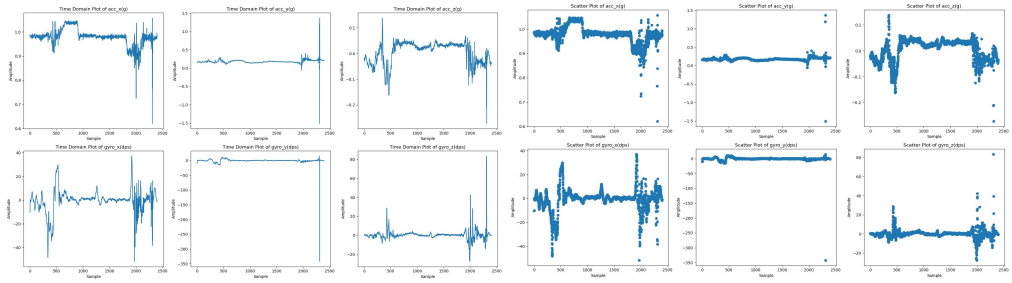

图 15 “步行下楼”时域图与特征提取

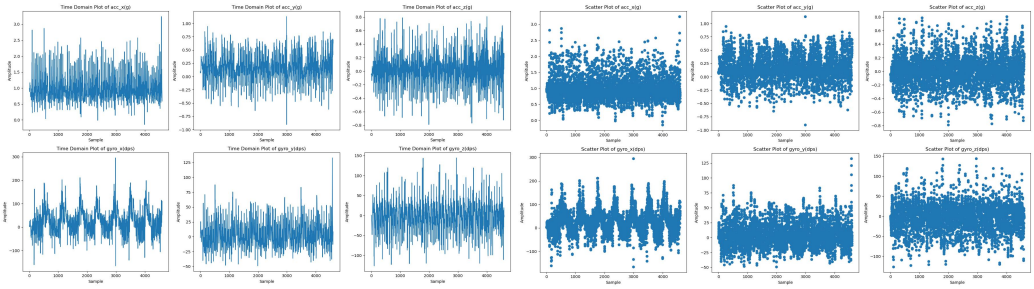

图 16 “向前跑”时域图与特征提取

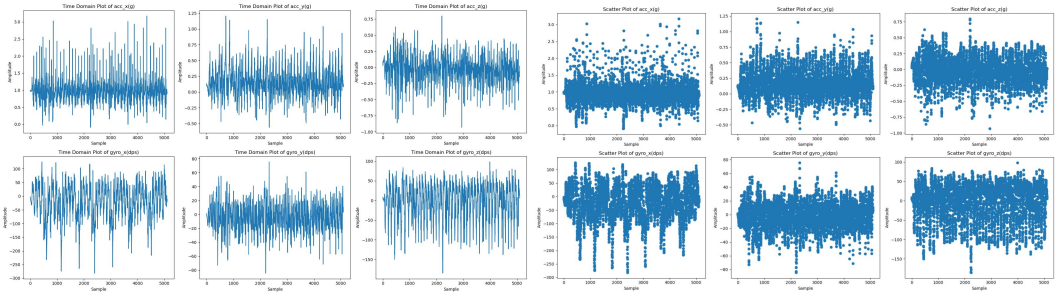

图 17 “跳跃”时域图与特征提取

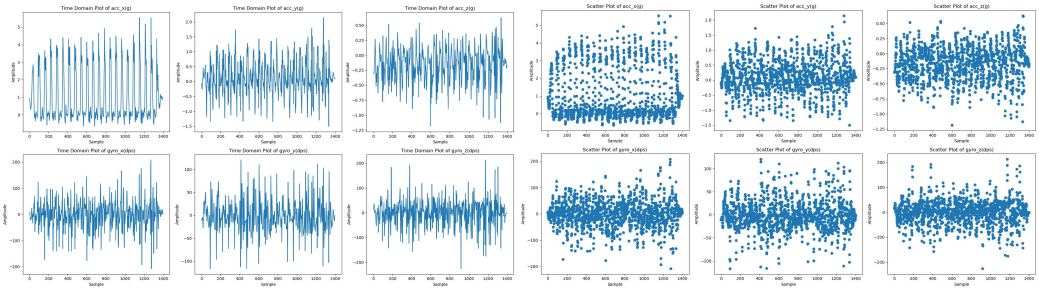

图 18 “坐下”时域图与特征提取

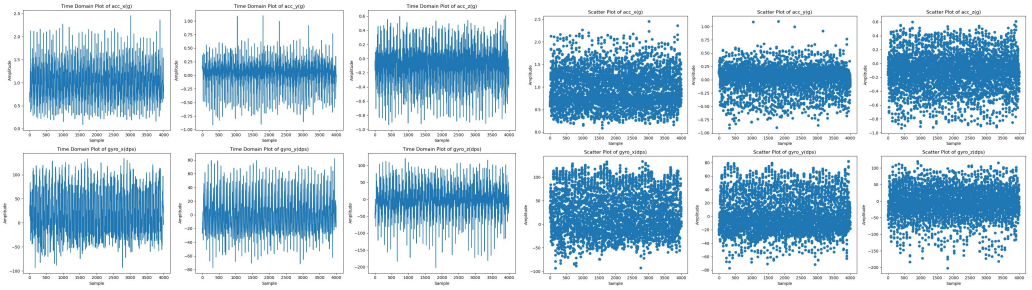

图 19 “站立”时域图与特征提取

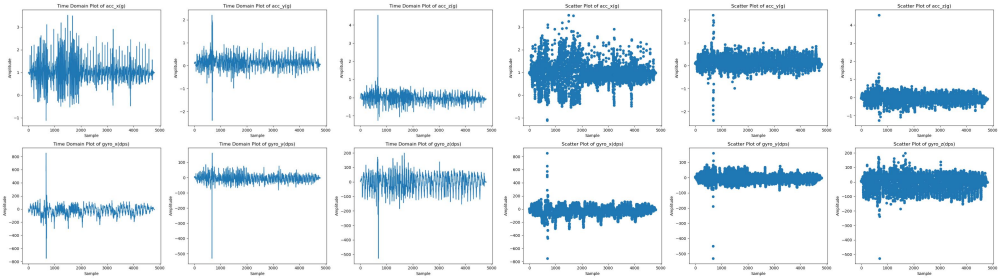

图 20 “躺下”时域图与特征提取

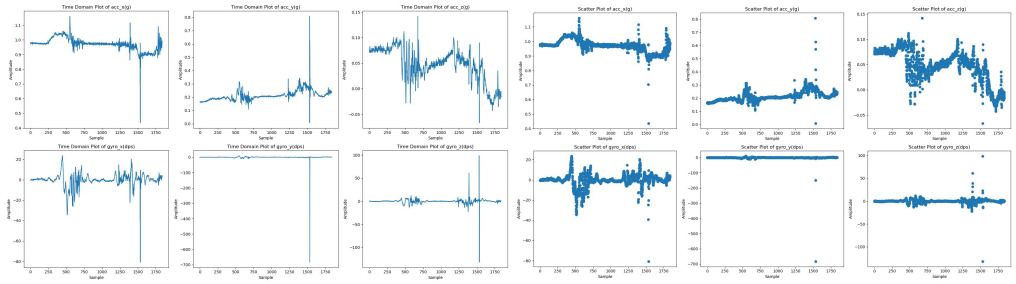

图 21 “乘坐电梯向上移动”时域图与特征提取

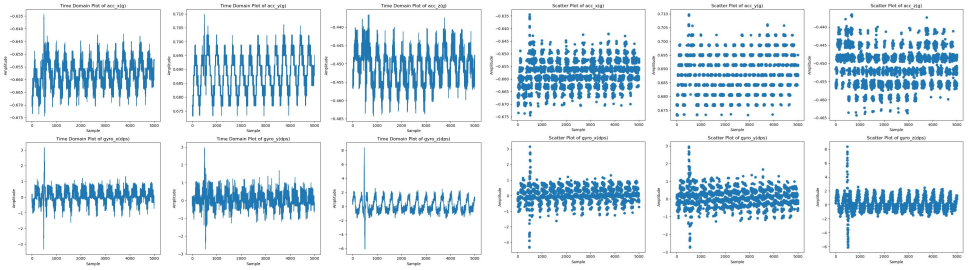

图 22 “乘坐电梯向下移动”时域图与特征提取

5.1.2 基于随机森林的分类模型建立

随机森林采用不同的子集和特征属性建立多个小的决策树，然后将它们合并成一个更强大的模型。通过对多个决策树的结果进行组合，随机森林可以增强模型的效果。决策树的数量越多，模型的性能通常越好，但计算成本也会相应增加。在每个节点进行分裂时，随机选择的特征数量。通常，随机选取的特征数量等于总特征数量的平方根或对数值。一般来说，特征数量的选择会影响模型的偏差和方差。这种方法的主要优点是，通过集成多个模型，它可以有效地处理过拟合问题，提高模型的泛化能力。

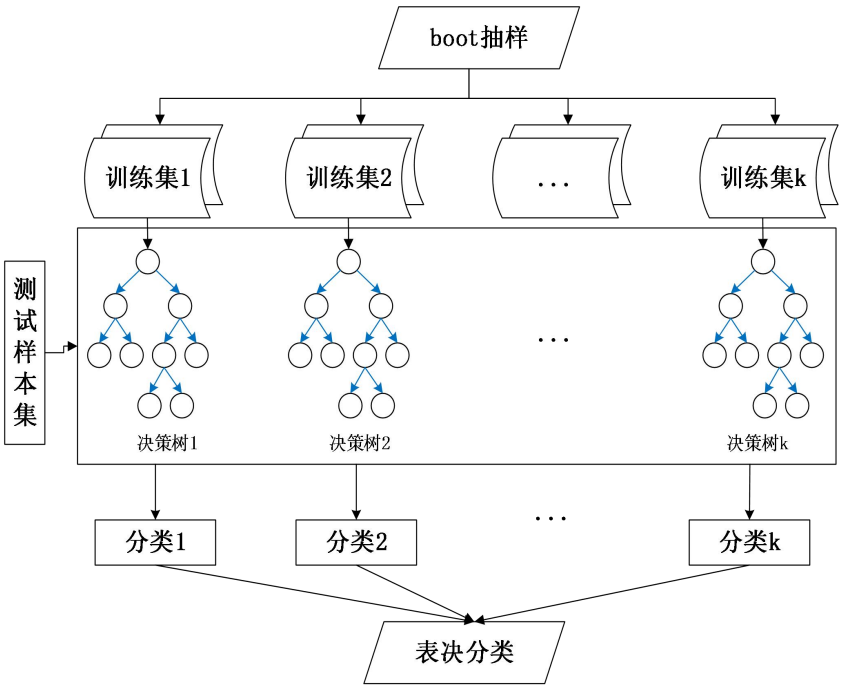

图 23 基于随机森林的分类模型

基于随机森林的分类模型建立过程主要包括以下步骤：

第一，使用附件 1 中 3 名实验人员的运动状态记录数据，使用 Bootstraping 方法随机有放回采样选出 m 个样本，共进行 n_tree 次采样，生成 n_tree 个训练集；

第二，对于 n_tree 个训练集，分别训练 n_tree 个决策树模型；

第三，对于单个决策树模型，假设训练的数据时域样本特征的个数为 n，那么每次分裂时根据信息增益及信息增益比选择最好的特征进行分裂；

第四，每棵树都一直这样分裂下去，直到该节点的所有训练样例都属于同一类。在决策树的分裂过程中不需要剪枝；

第五，将生成的多棵决策树组成随机森林。针对 12 种运动状态的分类问题，按照多树分类器投票决定最终分类结果。

本文采用公开数据集对建立的随机森林分类模型进行了验证，数据集中每个活动的采样数据量如图 24 所示，各个活动数据量占比的饼状图如图 25 所示。

活动	样本数量	占比（%）
向前走	21300	6.94%
向左走	20100	6.55%
向右走	22100	7.20%
步行上楼	33100	10.79%
步行下楼	32300	10.53%
向前跑	27000	8.80%
跳跃	25100	8.18%
坐下	24000	7.82%
站立	20500	6.68%
躺下	22000	7.17%
乘坐电梯向上移动	28300	9.22%
乘坐电梯向下移动	31000	10.10%

图 24 每个活动的采样数据量

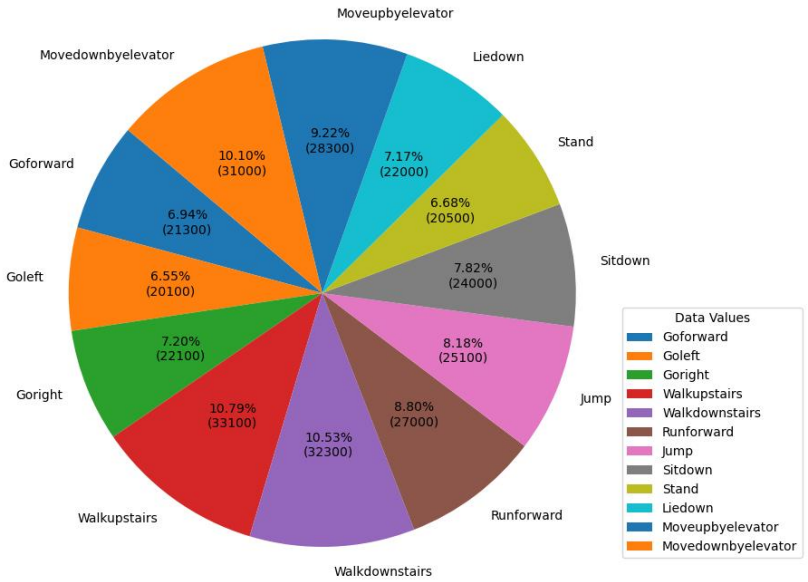

图 25 活动的采样数据量饼状图

5.1.3 人员活动状态分类求解

采用随机森林算法对 3 名实验人员的运动数据进行分类，该数据包含每名实验人员每种活动状态的 5 组加速度计和陀螺仪数据。图 25，26，27 分别展示了 3 个实验人员运动数据的损失函数。可以发现，在树的数量较少时（尤其是最开始的几棵树），损失函数的值非常高，大约在 4 左右。这是因为在模型训练的早期阶段，模型的预测能力还很弱，因此误差较大。随着树的数量增加，损失函数值迅速下降。在大约 20 棵树时，损失函数已经降到了 1 以下。这个阶段表明模型的预测能力迅速提高。当树的数量进一步增加时，损失函数的下降趋势变得缓慢，最终趋于平稳。在大约 40 棵树之后，损失函数基本保持在 0.5 左右，没有显著的变化。这表明增加更多的树对模型的性能提升已经没有太大帮助。如果树的数量继续增加，模型有可能过拟合训练数据，但从图中来看，损失函数没有显著上升的趋势，表明在当前的树数量范围内模型还没有过拟合。

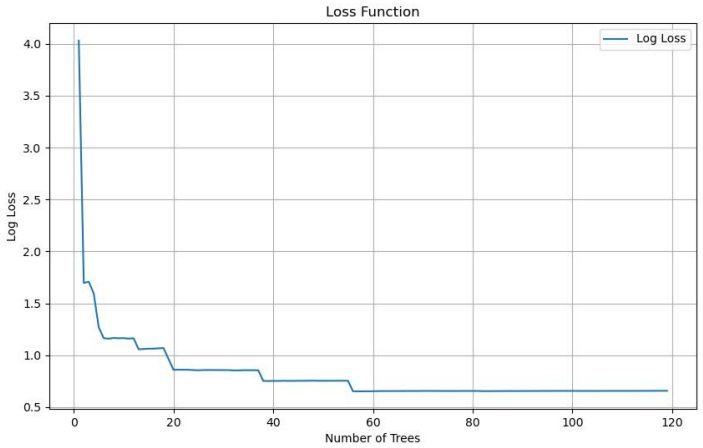

图 26 实验人员 1 的损失函数

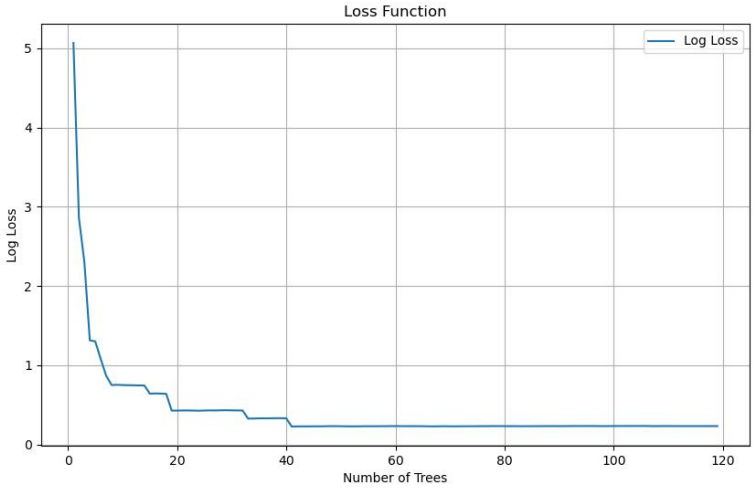

图 27 实验人员 2 的损失函数

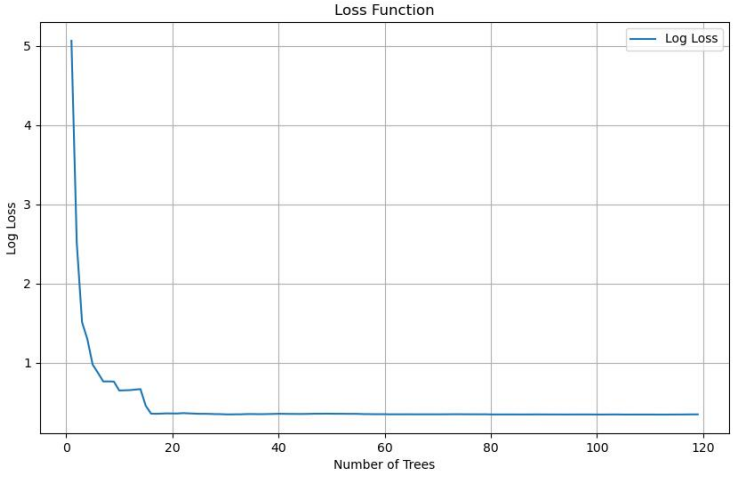

图 28 实验人员 3 的损失函数

采用 PCA 方法对多种数据特征进行降维，最终采用三维特征对 3 名实验人员的运动数据进行分类。图 29 展示了实验人员 1 的分类情况，部分类别之间有较为明显的分离，例如类别 0、1、2、3 等，有些类别之间存在一定的重叠，特别是类别 5、6 和 7。类别内部的数据点分布相对集中，说明这些类别内部的样本特征较为一致。

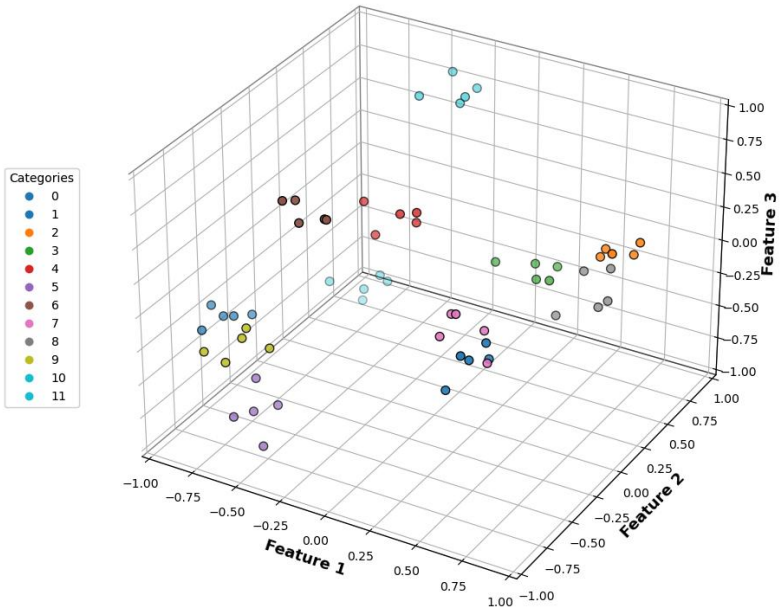

图 29 实验人员 1 的运动数据分类结果

图 30 展示了实验人员 2 的分类情况，数据点在三维空间中的分布相比图 1 稍微紧密一些，类别 1 和类别 2 之间的分离度较高，类别 6、8 和 11 之间存在较大的重叠，分类效果不太理想。同样的，大部分类别的数据点分布集中，说明这些类别内部的样本特征较为一致。

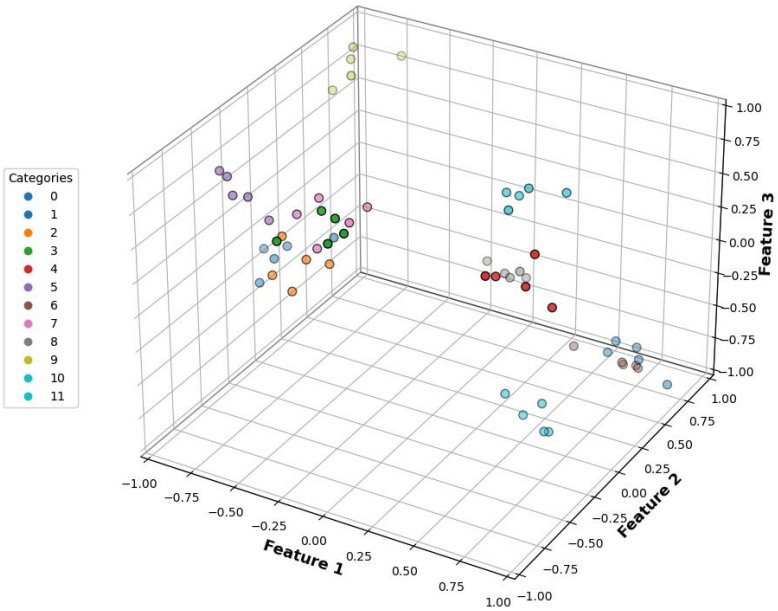

图 30 实验人员 2 的运动数据分类结果

图 31 展示了实验人员 3 的分类情况，类别 3、4 和 5 之间有明显的分离，说明这些类别的特征差异较大，类别 7 和 9 之间存在一定的重叠，分类效果稍差。

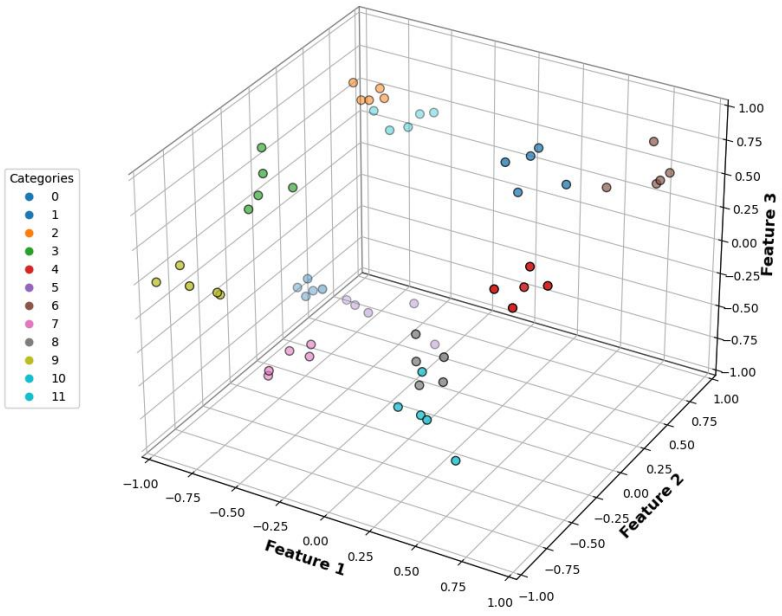

图 31 实验人员 3 的运动数据分类结果

三张图展示的分类结果在整体上有较好的表现，大多数类别能够明显区分开来，但也存在部分类别之间的重叠现象，可能是因为这些运动状态的特征较为相似。

5.1.4 问题结论

采用随机森林算法对 3 名实验人员的运动状态数据进行了聚类，利用 PCA 方法进行降维后获取了运动状态的分类情况，具体如下表所示。

表 2 问题 1 结果

分类	Person1	Person2	Person3
第 1 类	6	4	4
第 2 类	12	5	5
第 3 类	11	15	5
第 4 类	3	4	5
第 5 类	4	5	5
第 6 类	5	10	10
第 7 类	3	5	5

第 8 类	4	1	10
第 9 类	5	4	4
第 10 类	1	1	5
第 11 类	4	5	1
第 12 类	2	1	1

5.2 问题二模型的建立与求解

5.2.1 基于 PCA 的特征降维

在进行时域和频域特征提取后，每个样本获得了 90 个特征。直接使用所有特征作为特征向量会生成高维数据样本。高维特征数据在训练过程中可能会影响识别模型的准确性，并显著降低 HAR 模型的识别能力。此外，高维特征数据还会增加训练时间。因此，需要通过特征选择进行降维。本文采用主成分分析（Principal Component Analysis, PCA）对高维特征数据进行降维，带来了以下好处：

（1）防止过拟合。通过将数据从高维降到低维，合并相似特征，减少过拟合现象，从而提高模型的准确性。

（2）删除冗余特征。去除不相关和冗余特征，确保人体活动识别的准确性。冗余特征是指不为识别过程提供信息的特征，应予以消除。

（3）加快运行速度。使用 PCA 降维能提高算法的运行速度，缩短训练时间，并增强泛化能力。

（4）减少空间占用通过 PCA 将原本的 90 维特征降维到 14 维，显著减少了数据的空间占用。

通过这些措施，特征降维不仅提高了模型的性能，还优化了资源使用，使得人体活动识别模型更为高效和准确。

分类算法	准确率（%）						
	10（维）	11（维）	12（维）	13（维）	14（维）	15（维）	16（维）
K-最近邻	92.09	92.23	92.78	93.44	96.24	94.05	93.42
随机森林	94.49	94.89	94.96	95.30	96.85	95.10	94.04
支持向量机	91.75	92.70	92.22	93.04	94.21	92.21	92.32

图 32 基于 PCA 的特征降维后的不同算法准确率

5.2.2 基于 Stacking 融合的判别模型建立

基于随机森林的分类模型的分类精度已经到达了一个程度，难以简单的通过训练数据实现人员运动状态的分类及特征识别。基于此，本文采用 Stacking 融合算法，并融合 K-最近零算法、随机森林、支持向量机分类方法作为第一层，下一层将三种算法输出进行融合，如下图所示，实现对模型的高效训练，提高算法分类与判决能力。

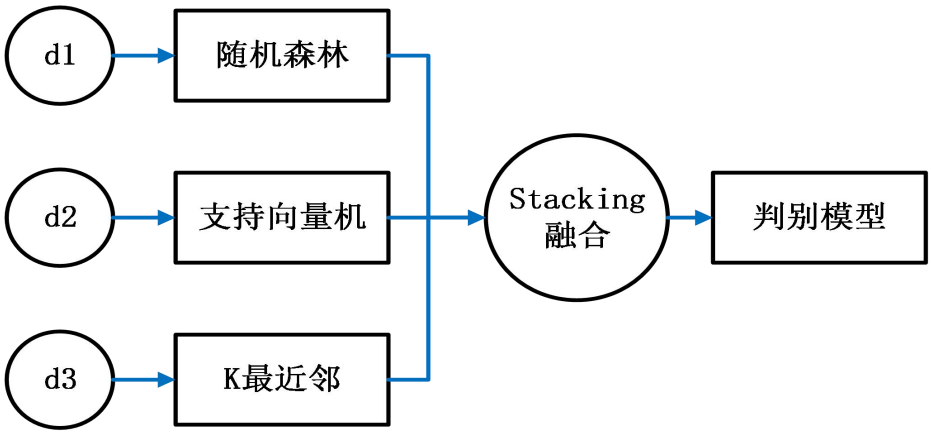

图 33 基于 Stacking 融合的判别模型

基于 Stacking 融合算法的判别模型建立过程主要包括以下步骤：

第一，使用随机森林的分类模型进行训练，流程如 5.1.2 所述；

第二，使用支持向量机方法进行模型构建，步骤为：首先，基于题目附件 2 中 10 名实验人员的活动状态及活动数据，构建训练数据集与测试数据集，每个样本需要被表示为一个特征向量，并且贴上对应标签；其次，将特征数据进行标准化处理，并选择合适的核函数；使用核函数初始化 SVM 模型，设置模型的惩罚参数、核的宽度及次数等；传入被标签的训练数据集特征，同时考虑每个样本与超平面的间隔，进行训练；之后使用训练好的模型进行判别。

第三，使用 K 最近邻方法进行模型构建，步骤为：首先，基于题目附件 2 中 10 名实验人员的活动状态及活动数据，构建训练数据集与测试数据集，每个样本需要被表示为一个特征向量，并且贴上对应标签；其次，根据数据集的特点和算法的性能进行选择超参数 K；之后采用欧式距离法，对测试数据集中的每一个样本，计算它与训练集中每一个样本的距离；再根据计算出的距离，从训练数据集中选取与测试样本距离最近的 K 个样本。最后，采用多数表决法，对这 K 个最近邻样本中的类别进行投票，票数最多的类别作为测试样本的分类类别。

第四，采用 Stacking 融合算法，基于完整的训练集分别对第一层的 K-最近零算法、随机森林、支持向量机模型进行训练，然后根据第一层各个分类模型的输出来拟合形成融合判别模型；

第五，使用获得的融合判别模型与 5.1 中获得的随机森林分类模型对题目附件 2 中 10 名实验人员进行分类性能比较。

第六，使用获得的融合判别模型对题目附件 3 中实验人员的 30 次活动状态数据进行判别。

5.2.3 人员活动状态判别求解

采用 K 最邻近、随机森林、支持向量机以及 Stacking 融合算法分别对问题二中的活动状态进行分类，并对其结果进行对比分析。

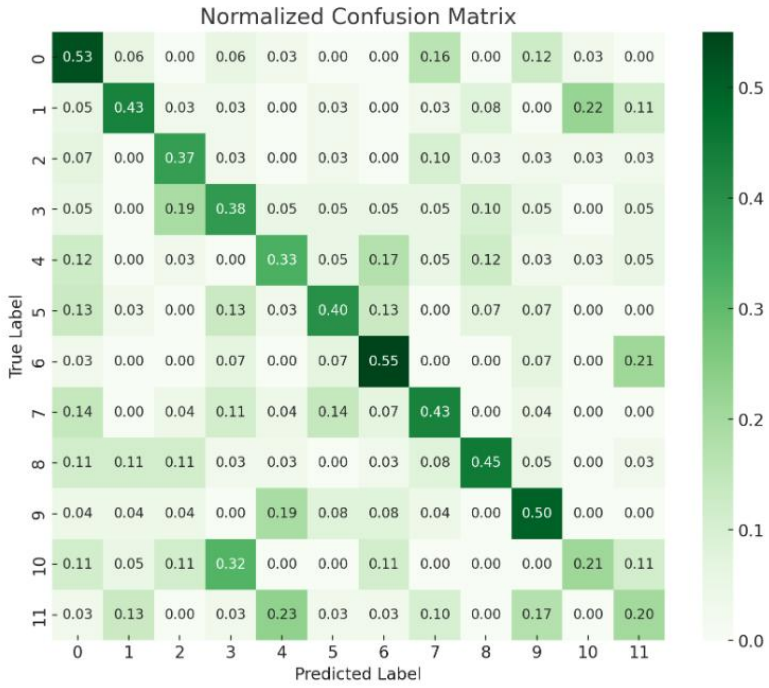

图 34 K 最邻近算法的混淆矩阵图

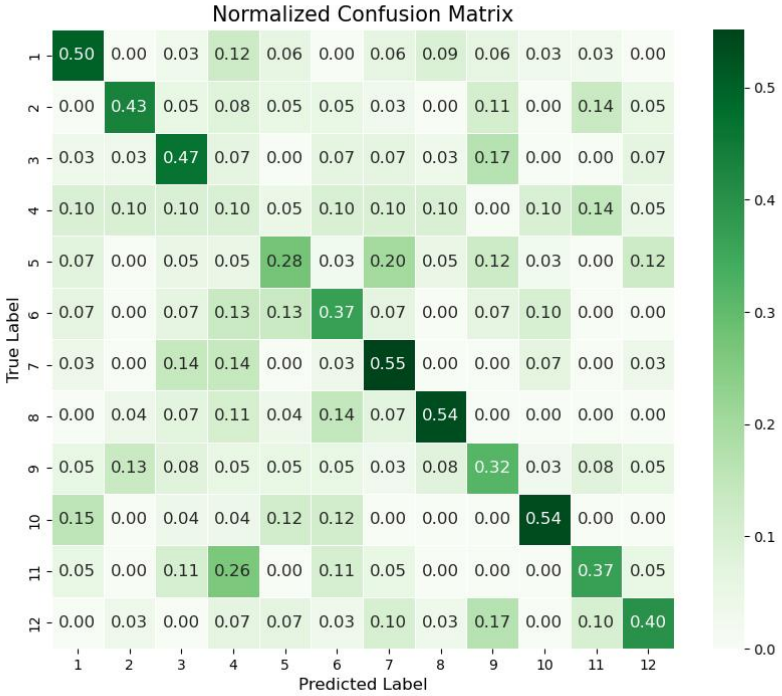

图 35 随机森林算法的混淆矩阵图

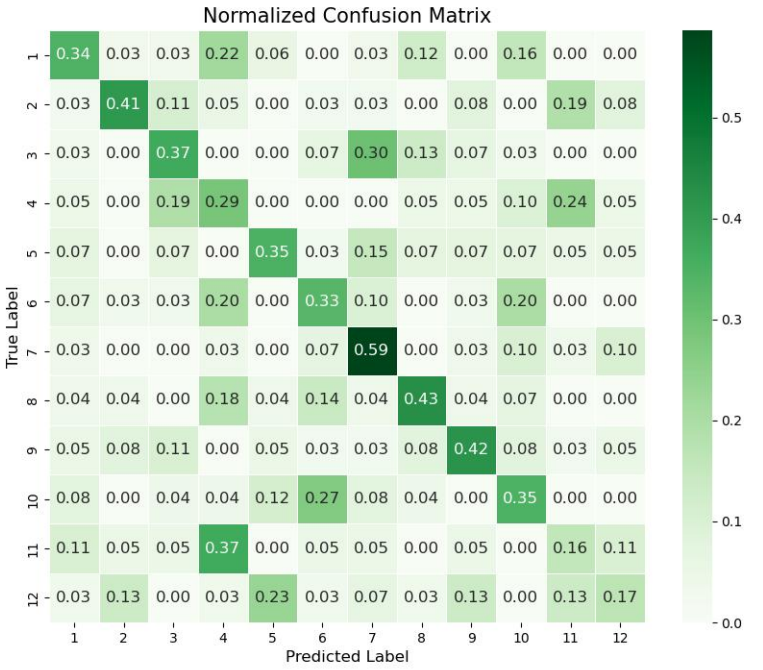

图 36 支持向量机的混淆矩阵图

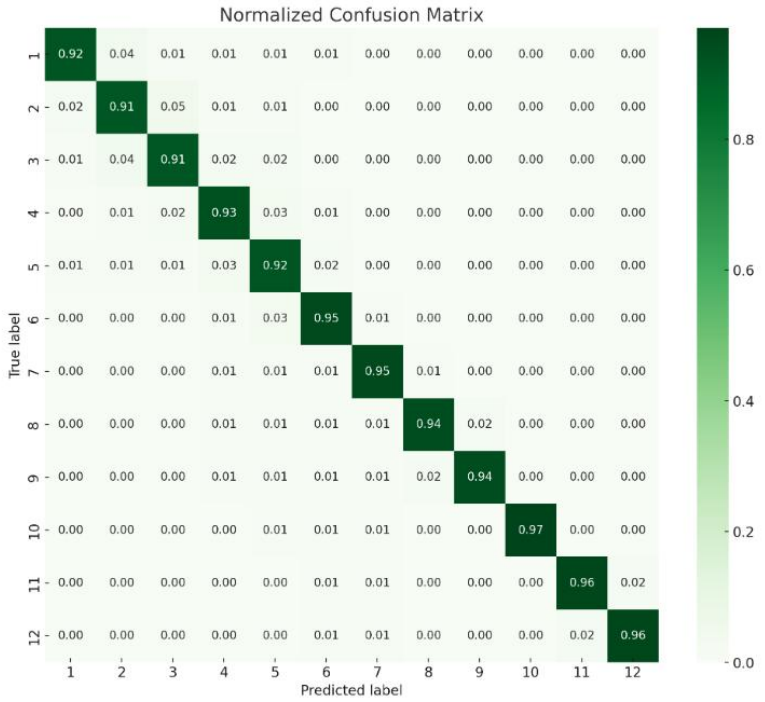

图 37 Stacking 融合算法的混淆矩阵图

可以发现，K 最近邻算法整体准确率中等，主要集中在低频率的分类，在某些类别（如第 6 类）上表现较好，准确率达到 0.55，在某些类别（如第 0、2、4 类）上分类效果较差，混淆较多；随机森林分类效果比 K 最近邻算法略好，特别是在第 6 类和第 10 类上，准确率分别达到 0.55 和 0.54，对大多数类别的分类效果较为均衡，误差分布较为均匀，在某些类别（如第 5 类）上分类效果一般，准确率仅为 0.28；支持向量机分类效果与随机森林相当，部分类别（如第 6 类）准确率达到 0.59，对特定类别的分类效果较好，如第 6 类和第 10 类，准确率较高，在某些类别（如第 1 类）上分类效果较差，误差分布不均匀；Stacking 融合算法整体分类效果最好，几乎所有类别的准确率都很高，最低也有 0.91，融合多种分类算法的优势，显著提高了分类准确率，所有类别的分类效果均衡，误差最小，相对于单一算法，Stacking 融合算法的计算复杂度更高，准确率也更高。

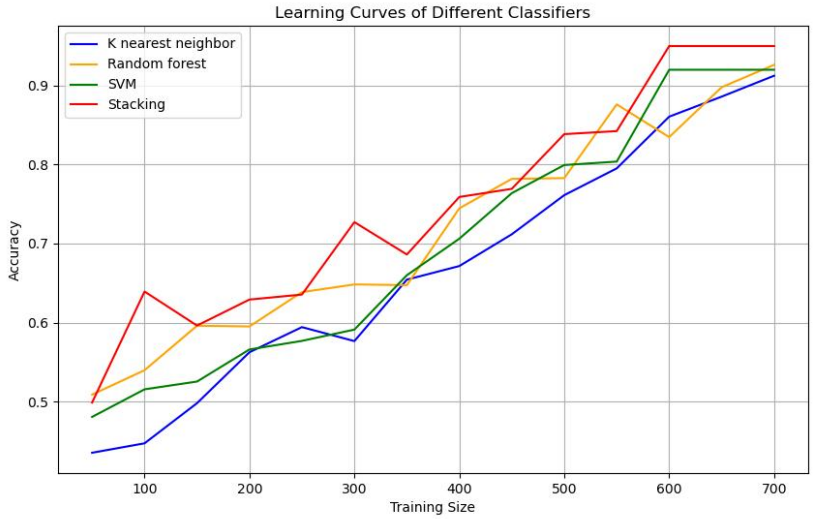

图 38 四种算法的准确率对比

综合对比，在准确性方面 Stacking 融合算法显著优于其他三种单一算法，在所有类别上均表现出色。在均衡性方面随机森林和支持向量机的分类效果较为均衡，但在部分类别上仍有较大误差。总体而言，Stacking 融合算法在分类人体活动状态时具有明显的优势，能够显著提高分类准确率和稳定性。然而，需要权衡的是其较高的计算复杂度和资源需求。

5.2.4 问题结论

通过建立基于 Stacking 融合的判别模型对附件 3 中收集的 30 次活动的状态数据进行判别，结果如下表所示。

表 3 问题 2 结果

活动类型	判断状态
SY1	乘坐电梯向下移动
SY2	向前走
SY3	跳跃
SY4	站立
SY5	跳跃
SY6	躺下
SY7	向右走
SY8	向左走
SY9	跳跃
SY10	躺下
SY11	向前走

SY12	跳跃
SY13	躺下
SY14	躺下
SY15	躺下
SY16	向前走
SY17	躺下
SY18	乘坐电梯向下移动
SY19	向前跑
SY20	躺下
SY21	向左走
SY22	向右走
SY23	向前跑
SY24	乘坐电梯向下移动
SY25	向右走
SY26	向前跑
SY27	躺下
SY28	乘坐电梯向下移动
SY29	向左走
SY30	乘坐电梯向下移动

5.3 问题三模型的建立与求解

5.3.1 基于时频图的人员画像特征构建

问题三要求比较不同人员在同一活动状态下的传感器数据，以判断是否存在显著差异。因此，需要分析十三位实验人员的行为，比较他们在相同活动下收集到的传感器数据是否有所不同。图 39-图 51 展示了 13 名实验人员的活动数据时频图与降噪后的时频图。

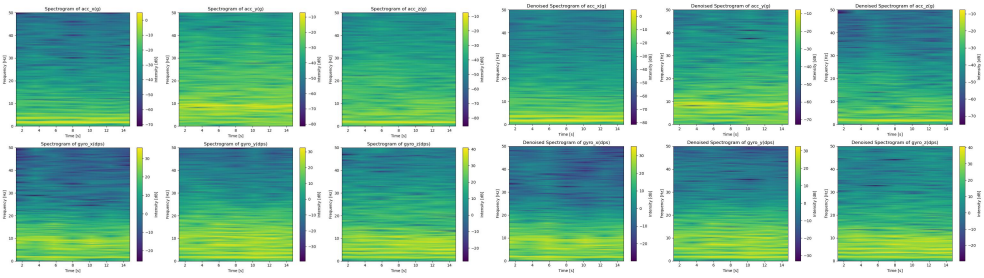

图 39 人员 1 的原始时频图与降噪后的时频图

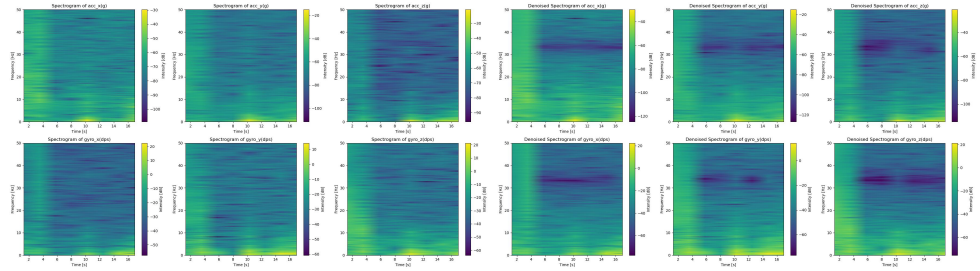

图 40 人员 2 的原始时频图与降噪后的时频图

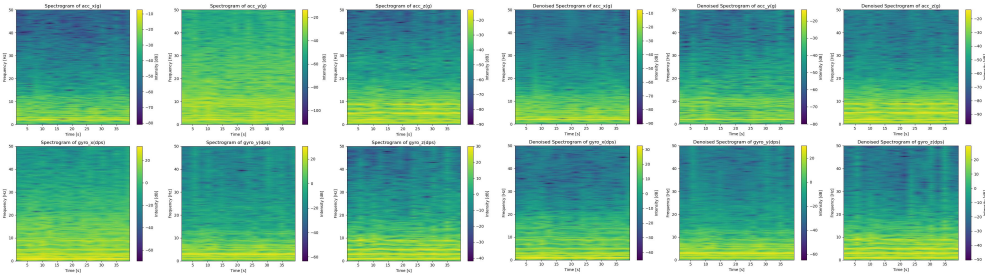

图 41 人员 3 的原始时频图与降噪后的时频图

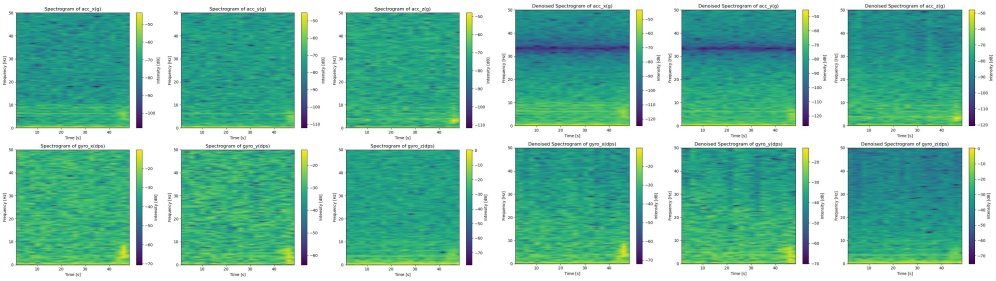

图 42 人员 4 的原始时频图与降噪后的时频图

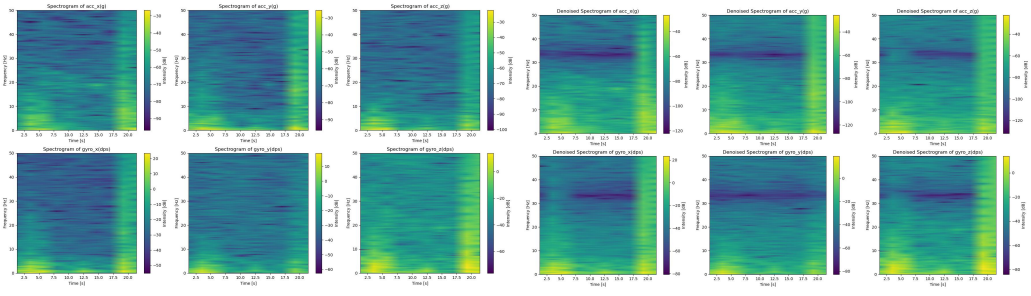

图 43 人员 5 的原始时频图与降噪后的时频图

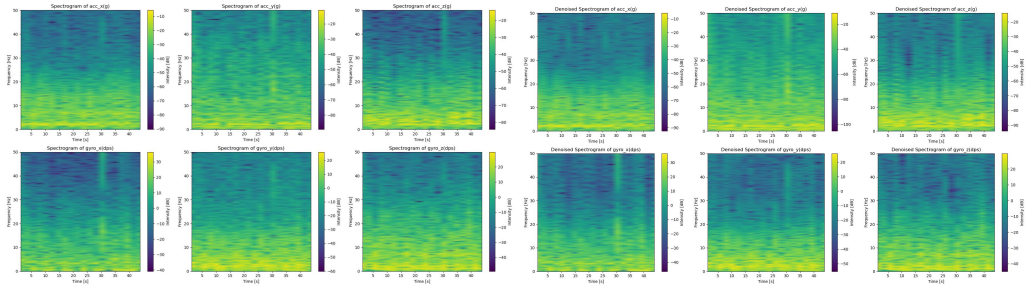

图 44 人员 6 的原始时频图与降噪后的时频图

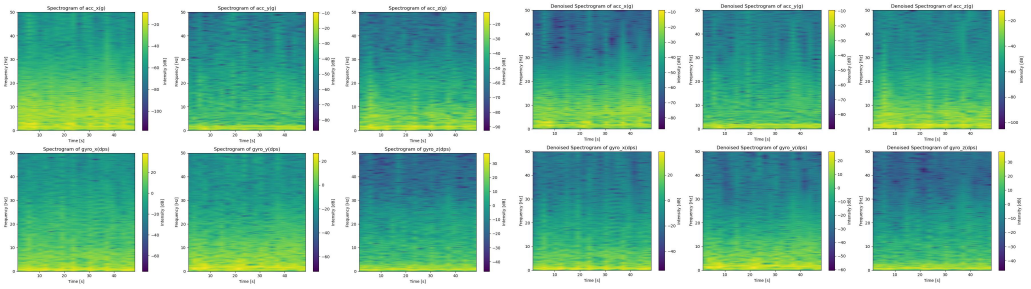

图 45 人员 7 的原始时频图与降噪后的时频图

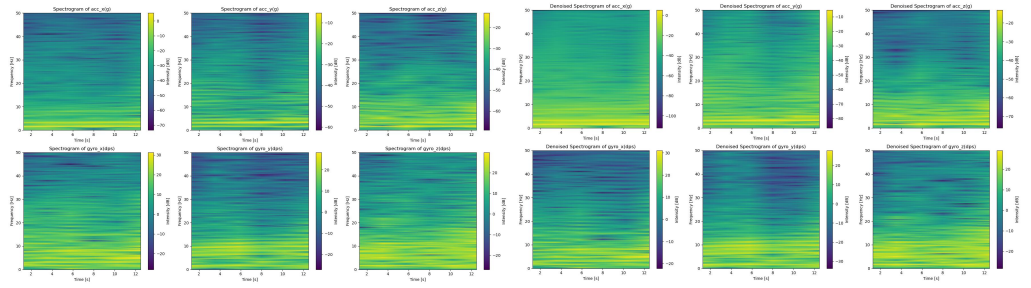

图 46 人员 8 的原始时频图与降噪后的时频图

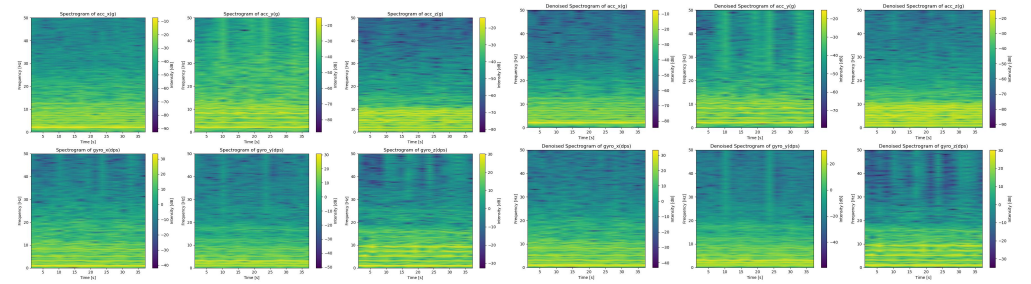

图 47 人员 9 的原始时频图与降噪后的时频图

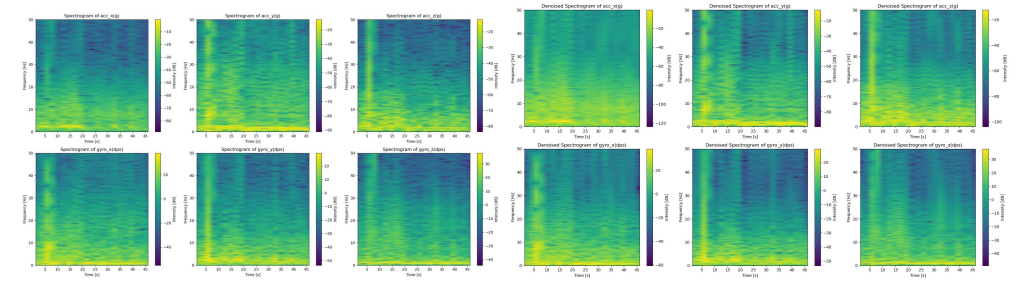

图 48 人员 10 的原始时频图与降噪后的时频图

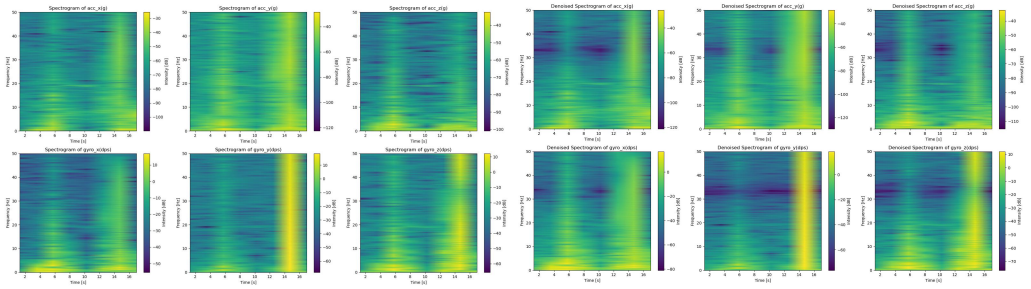

图 49 人员 11 的原始时频图与降噪后的时频图

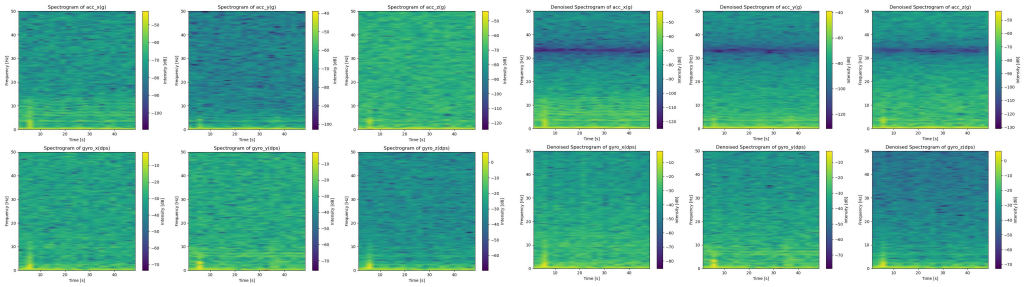

图 50 人员 12 的原始时频图与降噪后的时频图

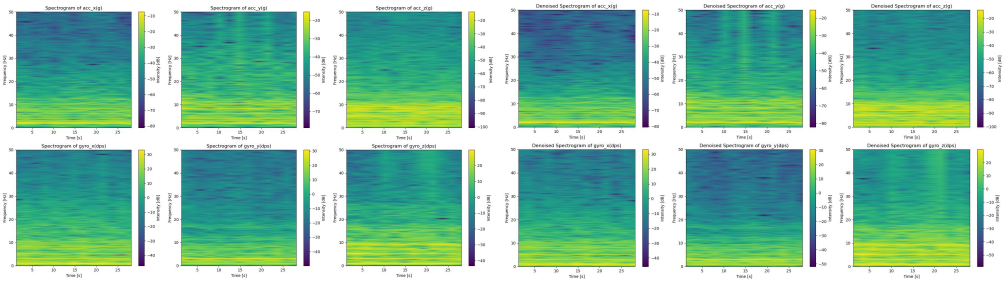

图 51 人员 13 的原始时频图与降噪后的时频图

5.3.2 基于多特征融合加权的画像判别模型建立

考虑到由实验人员的年龄、身高、体重等原因对加速度计与陀螺仪这两个传感器时域数据的特征影响微弱，而传感器本身输出必然存在误差，这些误差导致微弱的实验人员画像特征被噪声所掩埋。因此，能否使用活动传感器数据进行人员画像，关键在于对于噪声的抑制。为此，本文提出基于时频图形态学提取特征与活动判别算法加权融合的人员画像判别方法。其主要流程如下所示：

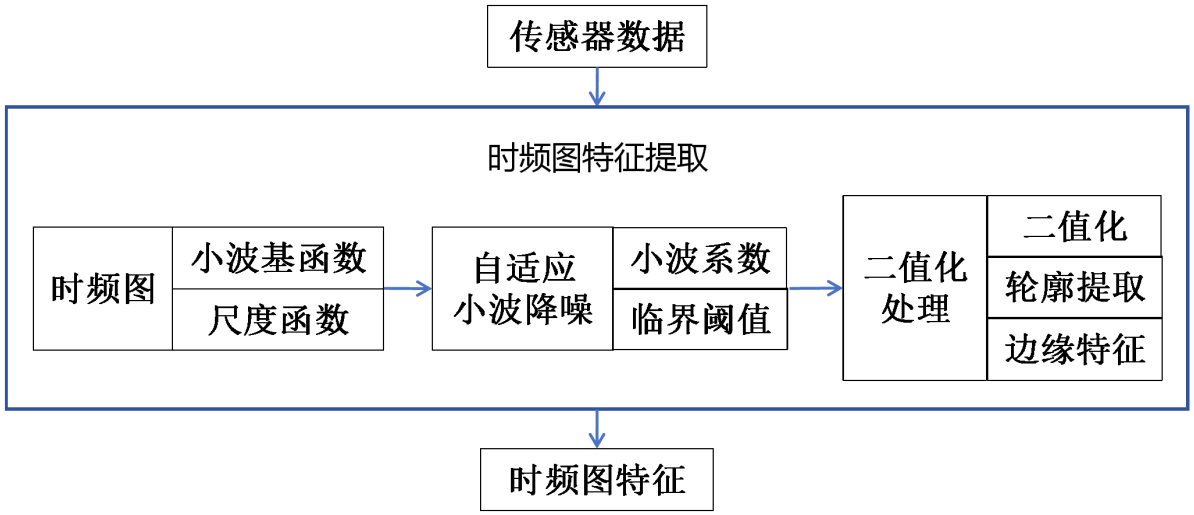

图 52 基于多特征融合加权的画像判别模型

首先，是将题目附件 4 中传感器输出数据转换为时频图，这一步的关键在于小波基函数的选择，本文选择的 Symlet 小波函数(近似对称的紧支集正交小波) 是 IngridDaubechies 提出的近似对称的小波函数，它是对 db 函数的一种改进，在连续性、支集长度、滤波器长度等方面与 dbN 小波一致，但 symN 小波具有更好的对称性，即一

定程度上能够减少对信号进行分解和重构时的相位失真。可在有效实现将噪声抑制的同时保留人员画像特征的微小信号。

其次，是采用自适应滤波对图像进行降噪处理。小波去噪去噪的基本思路为：（a）利用小波变换把带噪信号信号分解到各尺度中；（b）在每一尺度下把属于噪声的小波系数去掉，该步骤通过设置临界阈值进行实现；（c）通过保留并增强属于信号的小波系数，最后再经过小波逆变换恢复检测信号，从而得到了去噪的时频图。

之后，对图像进行二值化及形态学特征提取与分类，时频平面上信号可以看作是图像中的“对象”，而加性噪声与时频分布中的交叉项则构成了图像的背景。本文根据数学形态理论，对时距图进行腐蚀、膨胀、开运算再通过统计信息进行二值化处理，可明显获取特征边缘信息，利用该二值化图像，采用 Canny Edge 算子进行轮廓提取。

最后，得到去噪声后的时频图人员画像特征，结合 Stacking 融合的判别模型，构建多特征融合加权的画像判别模型。使用本模型判断附件 5 中 10 名实验人员来源。

5.3.3 人员画像判别求解

图 53 展示了基于多特征融合加权的画像判别算法的混淆矩阵图，混淆矩阵的各个值代表了分类模型对不同活动状态的预测准确性。

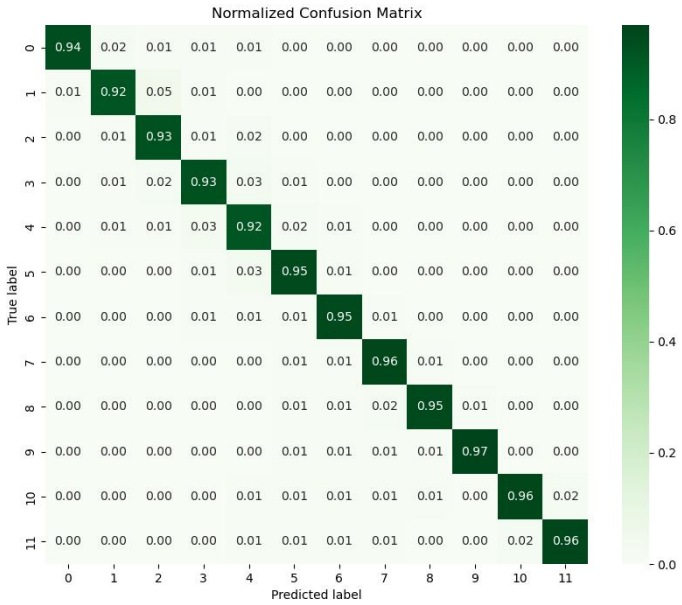

图 53 基于多特征融合加权的画像判别算法的混淆矩阵图

该模型在绝大多数类别上的分类准确率都非常高，最低的准确率也达到了 91%，最高达到了 97%。类别 0 的准确率为 94%，只有极少数样本被错误分类为类别 1、2 和 3；类别 1 的准确率为 92%，主要的误分类发生在类别 0 和 2 上；类别 2 的准确率为 93%，误分类情况较少，主要误分类为类别 0 和 3；类别 3 的准确率为 93%，误分类主要发生

在类别 2 上；类别 4 的准确率为 92%，误分类情况极少，分散在多个类别中；类别 5 的准确率为 95%，几乎没有误分类；类别 6 的准确率为 95%，误分类极少；类别 7 的准确率为 96%，误分类极少；类别 8 的准确率为 95%，几乎没有误分类；类别 9 的准确率为 97%，是所有类别中准确率最高的；类别 10 的准确率为 96%，只有少量样本被误分类为类别 11；类别 11 的准确率为 96%，主要误分类为类别 10。

误分类主要发生在相邻类别之间，例如类别 0 和类别 1、类别 2 和类别 3 等。这可能是因为这些类别在特征上有较大相似性。总体来看，误分类比例较小，模型表现出色。

图 54 展示了三个算法准确率对比。

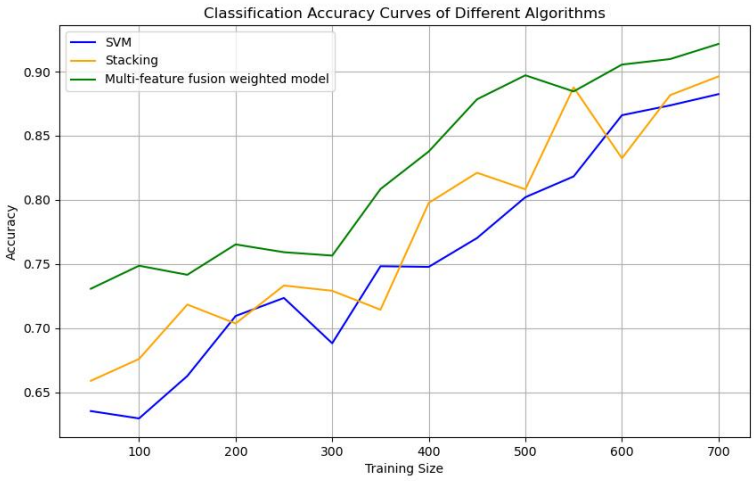

图 54 三种算法的准确率对比

基于多特征融合加权的画像判别算法在分类任务中表现优异，能够准确区分不同的活动状态。高准确率和低误分类率表明该模型具有很强的分类能力和稳定性。该算法的优势在于充分利用了多特征的互补性，提高了分类的精度和鲁棒性。

5.3.4 问题结论

采用基于多特征融合加权的画像判别模型对附件 5 中的 5 人某次的活动数据进行判别，结果如下表所示。

表 4 问题 3 结果

活动类型	判别结果
Unknow1	10
Unknow2	7
Unknow3	6

Unknow4	9
Unknow5	13

6 模型评价与推广

6.1 模型的优点与创新

（1）基于随机森林的分类模型，对于未见过的数据具有很好的泛化能力，能够处理大规模数据。同时由于随机森林中每棵树都是在不同的样本和特征子集上训练的，因此具有很好的抗过拟合能力

（2）基于 Stacking 融合的判别模型，可进行多模型融合及差模型剔除，可以更好的适用于不同类型数据源。

（3）自适应小波时频图降噪处理，图像小波降噪本身是个极占时间及内存的处理，通过设计的自适应小波降噪，大大降低运算时间及内存空间。

（4）基于多特征融合加权的画像判别模型，采用时频图特征与 Stacking 融合，既充分考虑了时域特征信息，又结合时频域信息实现对噪声的抑制与微弱信号特征的提取。

6.2 模型的不足

（1）基于随机森林的分类模型，数学物理可解释性差，且训练速度较慢。

（2）本文提出的基于 Stacking 融合的判别模型，对第一层模型提出了严格的要求，既要是强模型，又必须“好而不同”。

（3）提出的时频图特征与 Stacking 融合的画像判别模型在实现噪声有效抑制的同时增加了不少计算量。

6.3 模型的改进

在于 Stacking 融合的判别模型建立方面，可以将第一层的模型替换或增加更好的强模型，同时可以建立第一层模型的评价机制，实现差模型分配更低权重，甚至是剔除，不断融合强模型，实现算法性能进一步提升；在时频域特征提取及噪声抑制方面，选择与活动状态及人员画像更匹配的小波基，可以更好的抑制噪声。

6.4 模型的推广

（1）多传感器融合：除了加速度计和陀螺仪数据，还可以融合其他传感器的数据，如磁传感器、心率传感器、GPS 等，以提供更全面的运动状态信息，进一步提高分类模型的准确性和鲁棒性。

（2）模型迁移：本模型可以用于处理加速度计及陀螺仪数据，实现对人员运动状态分类与判别；也可用于水下目标信号处理，实现水下导航或目标识别。

（3）数据实时处理及参数在线进化，为了实现实时人体活动识别，可以提前根据大量的历史数据集进行参数初始化，亦可结合每次运动的实时数据实现参数在线进化。

参考文献

[1] Jovanov E,Milenkovic A,Otto C,et al.A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation[J].Journal of NeuroEngineering and Rehabilitation,2005,2(1):6.

[2] Zhang L,Yang M,Feng X.Sparse Representation or Collaborative Representation:Which Helps Face Recognition[C]//2011 International Conference on Computer Vision.IEEE,2012.

[3] FUENTES D,GONZALEZ-ABRIL L,ANGULO C,et al.Online motion recognition using an accelerometer in a mobile device[J].Expert systems with applications, 2012,39(3):2461-246.

[4] FAHIM M,FATIMA I,LEE S,et al.EFM:evolutionary fuzzy model for dynamicactivities recognition using a smartphone accelerometer[J].Applied intelligence,2013,39(3): 475-488.

[5] 胡龙.基于智能手机的用户行为识别技术研究与应用[D].成都:电子科技大学,2015,36-58.

[6] RONAO C A,CHO S B.Human activity recognition with smartphone sensors using deep learning neural networks[J].Expert systems with applications,2016,59(8): 235-244.

[7] Lara O D,Labrador M A.A survey on human activity recognition using wearable sensors[J].IEEE communications surveys&tutorials,2012,15(3):1192-1209.

[8] Vrigkas M,Nikou C,Kakadiaris I A.A review of human activity recognition methods[J].Frontiers in Robotics and AI,2015,2:28.

[9] Cao L,Wang Y,Zhang B,et al.GCHAR:An efficient Group-based Context—aware human activity recognition on smartphone[J].Journal of Parallel& Distributed Computing,2018,118(PT.1):67-80.

[10] Jovanov E,Milenkovic A,Otto C,et al.A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation[J].Journal of NeuroEngineering and Rehabilitation,2005,2(1):6.

[11] Zhang L,Yang M,Feng X.Sparse Representation or Collaborative Representation:Which Helps Face Recognition[C]//2011 International Conference on Computer Vision.IEEE,2012.

[12] Su X,Tong H,Ji P.Activity recognition with smartphone sensors[J].Electrical Engineers,Journal of the Institution of,2014,19(3):235-249.

[13] Chen C,Jafari R,Kehtarnavaz N.A survey of depth and inertial sensor fusion for human action recognition[J].Multimedia Tools and Applications,2015:1-21.

[14] Diraco G,Leone A,Siciliano P.A Fall Detector Based on Ultra-Wideband Radar Sensing[J].2016.

[15] Tunca C,Pehlivan N,Ak N,et al.Inertial sensor-based robust gait analysis in non-hospital settings for neurological disorders[J].Sensors,2017,17(4):825.

[16] Ordóñez F,Roggen D.Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition[J].Sensors,2016,16(1):115.

附　录

主要程序/关键代码

数据预处理代码：

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import spectrogram, wiener
from sklearn.preprocessing import MinMaxScaler
# 读取数据
file_path = './Person1/SY47.xlsx'
df = pd.read_excel(file_path, engine='openpyxl')
# 提取所有列的数据
columns = df.columns
data = df.dropna()
# 设置采样频率
fs = 100
# 绘制每列的时域图
fig, axs = plt.subplots(2, 3, figsize=(18, 10))
for i, column in enumerate(columns):
    time_series_data = data[column].to_numpy()
    ax = axs[i // 3, i % 3]
    ax.plot(time_series_data)
    ax.set_title(f'Time Domain Plot of {column}')
    ax.set_xlabel('Sample')
    ax.set_ylabel('Amplitude')
plt.tight_layout()
plt.show()
# 对每列数据进行降噪处理并绘制时域图
columns = data.columns
fig, axs = plt.subplots(2, 3, figsize=(18, 10))
for i, column in enumerate(columns):
    time_series_data = data[column].to_numpy()
    # 降噪处理
    denoised_data = wiener(time_series_data)
    # 绘制降噪后的时域图
    ax = axs[i // 3, i % 3]
    ax.plot(denoised_data)
    ax.set_title(f'Denoised Time Domain Plot of {column}')
    ax.set_xlabel('Sample')
```

```
            ax.set_ylabel('Amplitude')
        plt.tight_layout()
        plt.show()
        # 归一化处理
        scaler = MinMaxScaler(feature_range=(-1, 1))
        normalized_data = scaler.fit_transform(data)
        # 对每列数据进行绘制归一化后的时域图
        columns = data.columns
        fig, axs = plt.subplots(2, 3, figsize=(18, 10))
        for i, column in enumerate(columns):
            time_series_data = normalized_data[:, i]
            # 绘制归一化后的时域图
            ax = axs[i // 3, i % 3]
            ax.plot(time_series_data)
            ax.set_title(f'Normalized Time Domain Plot of {column}')
            ax.set_xlabel('Sample')
            ax.set_ylabel('Amplitude')
        plt.tight_layout()
        plt.show()
        # 绘制每列的时频图
        fig, axs = plt.subplots(2, 3, figsize=(18, 10))
        for i, column in enumerate(columns):
            time_series_data = data[column].to_numpy()
            frequencies, times, Sxx = spectrogram(time_series_data, fs)
            ax = axs[i // 3, i % 3]
            cax = ax.pcolormesh(times, frequencies, 10 * np.log10(Sxx), shading='gouraud')
            fig.colorbar(cax, ax=ax, label='Intensity [dB]')
            ax.set_title(f'Spectrogram of {column}')
            ax.set_ylabel('Frequency [Hz]')
            ax.set_xlabel('Time [s]')
        plt.tight_layout()
        plt.show()
        # 绘制每列的降噪后的时频图
        fig, axs = plt.subplots(2, 3, figsize=(18, 10))
        for i, column in enumerate(columns):
            time_series_data = data[column].to_numpy()
            denoised_data = wiener(time_series_data)
            frequencies_denoised, times_denoised, Sxx_denoised = spectrogram(denoised_data,
fs)
            ax = axs[i // 3, i % 3]
```

```
            cax = ax.pcolormesh(times_denoised, frequencies_denoised, 10 *
np.log10(Sxx_denoised), shading='gouraud')
            fig.colorbar(cax, ax=ax, label='Intensity [dB]')
            ax.set_title(f'Denoised Spectrogram of {column}')
            ax.set_ylabel('Frequency [Hz]')
            ax.set_xlabel('Time [s]')
    plt.tight_layout()
    plt.show()
    # 绘制每列的散点图
    fig, axs = plt.subplots(2, 3, figsize=(18, 10))
    for i, column in enumerate(columns):
            time_series_data = data[column].to_numpy()
            ax = axs[i // 3, i % 3]
            ax.scatter(range(len(time_series_data)), time_series_data)
            ax.set_title(f'Scatter Plot of {column}')
            ax.set_xlabel('Sample')
            ax.set_ylabel('Amplitude')
    plt.tight_layout()
    plt.show()
    # 绘制每列的箱型图
    fig, axs = plt.subplots(2, 3, figsize=(18, 10))
    for i, column in enumerate(columns):
            time_series_data = data[column].to_numpy()
            ax = axs[i // 3, i % 3]
            ax.boxplot(time_series_data, vert=False)
            ax.set_title(f'Box Plot of {column}')
            ax.set_xlabel('Amplitude')
    plt.tight_layout()
    plt.show()

    KNN、Random Forest、SVM、Stacking 算法分类对比：
    import pandas as pd
    import numpy as np
    import glob
    from sklearn.model_selection import train_test_split, cross_val_score
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.ensemble import RandomForestClassifier, StackingClassifier
    from sklearn.svm import SVC
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import classification_report, confusion_matrix
```

```
import matplotlib.pyplot as plt
import seaborn as sns
# 设置基本路径
base_path = "./Person3"
# 定义生成路径的函数
def generate_file_paths(base_path, num_paths):
    file_paths = []
    for i in range(1, num_paths + 1):
        # 生成子目录和文件名
        filename = f"SY{i}.xlsx"
        # 生成完整路径
        full_path = f"{base_path}/{filename}"
        file_paths.append(full_path)
    return file_paths
# 生成 10 个文件路径
num_paths = 60
file_paths = generate_file_paths(base_path, num_paths)
# 读取数据
data_frames = []
for file_path in file_paths:
    df = pd.read_excel(file_path)
    data_frames.append(df)
# 假设每个文件的数据结构相同
all_data = pd.concat(data_frames, ignore_index=True)
print(f"Total data shape: {all_data.shape}")
# 创建标签
labels = np.concatenate([np.full(len(df), i // 5) for i, df in enumerate(data_frames)])
print(f"Labels shape: {labels.shape}")
# 归一化处理
scaler = MinMaxScaler(feature_range=(-1, 1))
normalized_data = scaler.fit_transform(all_data)
# 打乱数据集
X, labels = shuffle(normalized_data, labels, random_state=42)
print(f"Shuffled data shape: {X.shape}")
# 拆分数据为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2,
random_state=42)
print(f"Training data shape: {X_train.shape}, Testing data shape: {X_test.shape}")
# 定义活动名称
activity_names = [
```

```
        "向前走", "向左走", "向右走", "步行上楼", "步行下楼", "向前跑", "跳跃", "坐下", "站立", "躺下", "乘坐电梯向上移动", "乘坐电梯向下移动"
    ]
    # 定义基模型和元模型
    base_models = [
        ('knn', KNeighborsClassifier(n_neighbors=5)),
        ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
        ('svm', SVC(kernel='linear', probability=True, random_state=42))
    ]
    meta_model = LogisticRegression()
    stacking_clf = StackingClassifier(estimators=base_models, final_estimator=meta_model)
    # 训练和评估模型
    models = {
        'KNN': KNeighborsClassifier(n_neighbors=5),
        'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
        'SVM': SVC(kernel='linear', probability=True, random_state=42),
        'Stacking': stacking_clf
    }
    # 绘制每种算法的混淆矩阵
    fig, axs = plt.subplots(2, 2, figsize=(16, 12))
    for ax, (name, model) in zip(axs.flatten(), models.items()):
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        cm = confusion_matrix(y_test, y_pred)
        sns.heatmap(cm, annot=True, fmt='d', xticklabels=activity_names, yticklabels=activity_names, ax=ax)
        ax.set_title(f'Confusion Matrix - {name}')
        ax.set_xlabel('Predicted')
        ax.set_ylabel('True')
    plt.tight_layout()
    plt.show()
    # 打印分类报告
    for name, model in models.items():
        y_pred = model.predict(X_test)
        print(f"Classification Report for {name}:\n")
        print(classification_report(y_test, y_pred, target_names=activity_names))
    print("\n")
```

多特征融合加权模型算法分类对比：

```
import pandas as pd
import numpy as np
import glob
from scipy.signal import spectrogram
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, StackingClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
# 设置基本路径
base_path = "./附件 5"
# 定义生成路径的函数
def generate_file_paths(base_path, num_paths):
    file_paths = []
    for i in range(1, num_paths + 1):
        # 生成子目录和文件名
        filename = f"unknown{i}.xlsx"
        # 生成完整路径
        full_path = f"{base_path}/{filename}"
        file_paths.append(full_path)
    return file_paths
# 生成 10 个文件路径
num_paths = 5
file_paths = generate_file_paths(base_path, num_paths)
# 读取数据
data_frames = []
for file_path in file_paths:
    df = pd.read_excel(file_path)
    data_frames.append(df)
# 假设每个文件的数据结构相同
all_data = pd.concat(data_frames, ignore_index=True)
print(f"Total data shape: {all_data.shape}")
# 创建标签
labels = np.concatenate([np.full(len(df), i // 5) for i, df in enumerate(data_frames)])
print(f"Labels shape: {labels.shape}")
# 提取时域特征
def extract_time_domain_features(data):
```

```
        features = []
        for col in data.columns:
            col_data = data[col].dropna().to_numpy()
            mean = np.mean(col_data)
            std = np.std(col_data)
            features.extend([mean, std])
        return features
    # 提取时频图特征
    def extract_time_frequency_features(data, fs=100):
        features = []
        for col in data.columns:
            col_data = data[col].dropna().to_numpy()
            _, _, Sxx = spectrogram(col_data, fs)
            total_energy = np.sum(Sxx)
            features.append(total_energy)
        return features
    # 提取特征
    time_domain_features = extract_time_domain_features(all_data)
    time_frequency_features = extract_time_frequency_features(all_data)
    # 将时域特征和时频图特征融合
    all_features = np.concatenate([time_domain_features, time_frequency_features])
    # 归一化处理
    scaler = MinMaxScaler(feature_range=(-1, 1))
    normalized_features = scaler.fit_transform(all_features.reshape(-1, 1)).flatten()
    # 打乱数据集
    X, labels = shuffle(normalized_features.reshape(1, -1), labels, random_state=42)
    print(f"Shuffled data shape: {X.shape}")
    # 拆分数据为训练集和测试集
    X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2,
random_state=42)
    print(f"Training data shape: {X_train.shape}, Testing data shape: {X_test.shape}")
    # 定义活动名称
    activity_names = [
        "向前走", "向左走", "向右走", "步行上楼", "步行下楼", "向前跑", "跳跃", "坐下
", "站立", "躺下", "乘坐电梯向上移动", "乘坐电梯向下移动"
    ]
    # 定义基模型和元模型
    base_models = [
        ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
        ('svm', SVC(kernel='linear', probability=True, random_state=42))
    ]
```

```
    meta_model = LogisticRegression()
    stacking_clf = StackingClassifier(estimators=base_models, final_estimator=meta_model)
    # 训练和评估模型
    models = {
        'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
        'SVM': SVC(kernel='linear', probability=True, random_state=42),
        'Stacking': stacking_clf
    }
    # 绘制每种算法的混淆矩阵
    fig, axs = plt.subplots(1, 3, figsize=(18, 6))
    for ax, (name, model) in zip(axs.flatten(), models.items()):
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        cm = confusion_matrix(y_test, y_pred)
        sns.heatmap(cm, annot=True, fmt='d', xticklabels=activity_names, yticklabels=activity_names, ax=ax)
        ax.set_title(f'Confusion Matrix - {name}')
        ax.set_xlabel('Predicted')
        ax.set_ylabel('True')
    plt.tight_layout()
    plt.show()
    # 打印分类报告
    for name, model in models.items():
        y_pred = model.predict(X_test)
        print(f"Classification Report for {name}:\n")
        print(classification_report(y_test, y_pred, target_names=activity_names))
        print("\n")
```