基于穿戴装备的身体活动监测

摘要

便携式可穿戴设备（如智能手环、智能手表等）的普及，极大地扩展了健康行为数据的采集规模，为医学应用带来了全新的机遇。微机电系统（MEMS）加速度计因其低成本、低功耗和便捷性，成为识别和监测身体活动的重要工具。

随着海量数据的不断积累，如何高效且精准地分析和利用这些数据已成为一项重大挑战。为应对这一问题，本文将基于**大数据处理框架 Polars**，结合智能技术与先进的数据分析方法，进行深入分析与求解。

针对问题一，需对 100 位志愿者的加速度记录数据（约 95.86 GB）进行活动信息统计。其中数据“annotation”列中存在缺失值，且这些缺失值通常出现在两个活动状态之间（除了 P052 在结尾处有一段缺失值）。因此，假设缺失值区间代表前后两种活动状态之间的“转变区”，并采用均值填充法进行 MET 值填补。数据填补后，结合多进程和 Polars 的方法——**Multi-Polars**，仅耗时 **51.31 秒**完成统计汇总任务。

针对问题二，根据加速度数据区间的规律，对非空区间的加速度数据进行了**统计特征**和**频域特征**的提取，构建了一个特征数据集。随后，使用 **XGBoost** 模型进行建模，得到的模型 **WAPE** 为 **0.2895**，**MSE** 为 **0.6032**。采用 **10000 个**数据点作为时间窗口对加速度数据进行预测。

然而，由于数据仅在开头和结尾处呈现睡眠状态，且模型无法识别这一特点，特此设计了一个**睡眠匹配算法**。该算法通过根据年龄、性别、记录开始时间和结束时间，将附件 2 中的志愿者与附件 1 中的志愿者进行匹配，将匹配得到的睡眠数据写入附件 2 中。最后，进行志愿者运动强度信息的统计，统计结果见表 8。

针对问题三，结合前述分析可知，睡眠状态分布在记录的开始和结束时间，因此需要分别对这两个时间段进行睡眠阶段的识别。通过对睡眠阶段数据的可视化，发现当 MET 值发生**显著变化**时，整体 MET 值会稳定在一个新 MET 值上。因此，将这一变化视为睡眠阶段的转变点。结合**差分**和**自适应阈值**算法，能够有效地识别睡眠阶段的变化，识别结果见表 9。

针对问题四，久坐行为被定义为连续 30 分钟内 MET 值小于 1.6 的状态。在问题二中，已将数据区间设置为 10000 个数据点，而 30 分钟对应 18 个区间。因此，采用 18 个区间的**滑动窗口算法**对久坐行为进行实时预警，预警结果见表 10。

关键词：Polars；多进程；XGBoost；睡眠匹配；差分；自适应阈值；滑动窗口

目录

一、引言

在数字健康时代，便携可穿戴设备（如智能手环、智能手表等）的普及率呈指数级增长，推动了健康行为数据采集进入了规模化和实时化的新阶段。微机电系统（MEMS）加速度计凭借其低成本、低功耗和高便携性的优势，成为捕捉人体运动轨迹与生理特征的核心传感器，为医疗健康领域提供了前所未有的数字化研究平台。然而，伴随着数据量的爆发性增长，如何实现高效的数据分析和精准建模，成为了一项巨大的挑战。

其中，代谢当量（MET）作为衡量运动强度的国际标准，以基础代谢率（1 MET）为基准划分运动强度等级，在个性化健康指导和运动处方制定中发挥着至关重要的作用。如何基于加速度计数据构建高泛化性的机器学习模型，实现个体 MET 值的实时精准估计，成为突破健康行为量化瓶颈的关键。

与此同时，公众对睡眠健康的关注度日益攀升，加速度计作为一种非侵入式的监测手段，为睡眠状态分析提供了便捷的解决方案。开发能够自动识别浅睡、深睡和快速眼动（REM）等睡眠阶段的算法，不仅有助于揭示睡眠的生理机制，更能为改善睡眠质量提供科学依据。此外，长时间久坐已被证实为心血管疾病、糖尿病等慢性疾病的重要风险因素。基于加速度计数据自动识别 MET 值低于 1.6 且持续超过 30 分钟的久坐行为，并实现实时预警，对促进主动健康管理、降低疾病风险具有重要的现实意义。

本文将建立以下四个模型对相关问题进行分析与求解：

1）基于 Multi-Polars 的 MET 值统计模型

2）基于 StatXBoost 的 MET 值估计模型

3）基于差分与自适应阈值的睡眠阶段识别模型

4）基于滑动窗口的久坐行为健康预警模型

二、基于 Multi-Polars 的 MET 值统计模型

2.1 问题分析

问题：根据加速度记录数据，统计汇总附件 1 中各个志愿者的身体活动信息，数值保留小数点后 4 位。

分析：附件 1 中共包含了 100 位志愿者的加速度记录数据，单个数据（.csv）约 1GB 存储空间，平均包含量级10^7（约 1000 万条）的采样记录。需要注意的是，数据中的“annotation”列存在缺失值。

因此，该问题需要解决两个关键点：**1）如何有效处理缺失数据；2）如何设计一种高效的统计汇总方案。**

2.2 Multi-Polars 模型

截至目前，Pandas 一直是数据处理和分析领域的主流工具。然而，在处理大规模数据集时，Pandas 常常面临性能瓶颈，导致内存和计算资源的过度消耗。在这种情况下，Polars 作为一种高效的解决方案，能够显著提升数据处理效率，并有效减少资源消耗。

Polars 是一个高性能的数据处理库，专为大规模数据处理而设计[1]。它提供了类似 Pandas 的 DataFrame 操作，但在性能上通过并行计算和内存优化实现了显著提升。Polars 采用 Rust 编写，充分利用现代多核处理器的优势，提供更高的计算速度和更低的内存占用。与 Pandas 相比，Polars 在处理大型数据集时展现出了卓越的性能，特别是在多线程计算和内存优化方面具有明显优势。与 Dask 相比，Polars 在本地单机环境下表现尤为突出。

多进程是一种通过创建多个独立进程来实现并行处理的技术。每个进程拥有独立的内存空间和资源，因此进程之间的数据不会直接共享，避免了线程间的竞争和资源冲突。多进程能够有效利用多核处理器，特别适用于 CPU 密集型任务，从而提高计算速度和资源利用率。

本文提出了一种结合多进程与 Polars（Multi-Polars）的高效处理方案。该方案首先构建多个进程，并在每个进程中利用 Polars 对单个文件进行高效的数据处理与统计汇总，最后将各文件的处理结果进行整合与汇总。通过充分发挥 Polars 在数据处理方面的高效性以及多进程在并行计算中的优势，显著提升了整体处理效率。

Multi-Polars 模型流程图如图 1 所示：

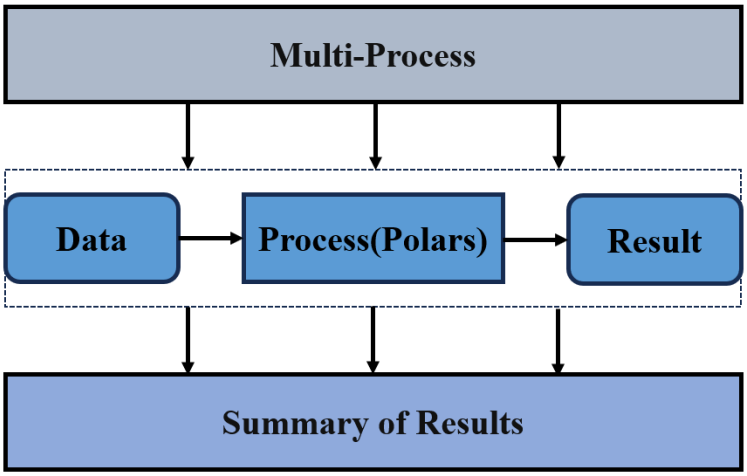

图 1 Multi-Polars 模型流程图

Figure 1 Flow chart of the Multi-Polars Model

2.3 理论速度测试

为对比不同方案在数据处理速度上的表现，本文设置了以下实验环境和测试内容，具体如下：

（1）实验环境：

CPU	AMD Ryzen 5 5600 6-Core Processor
显卡	AMD Radeon RX 6500 XT
内存	Crucial DDR4 3200 (16GB*2)
固态	TiPlus 7100 Gen4 512GB

注：该配置处于较低水平

（2）实验内容 1：

使用 Pandas、Dask 和 Polars 库，分别对附件一中志愿者的加速度记录数据进行检索测试。针对数据列“annotation”中值为“7030 sleeping; MET 0.95”的记录，执行检索操作，并计算平均检索时间。

在检索任务相同的情况下，运行速度V可视为运行时间T的倒数，即：

$$V = \frac{1}{T} \tag{1}$$

实验结果如图 2 所示：

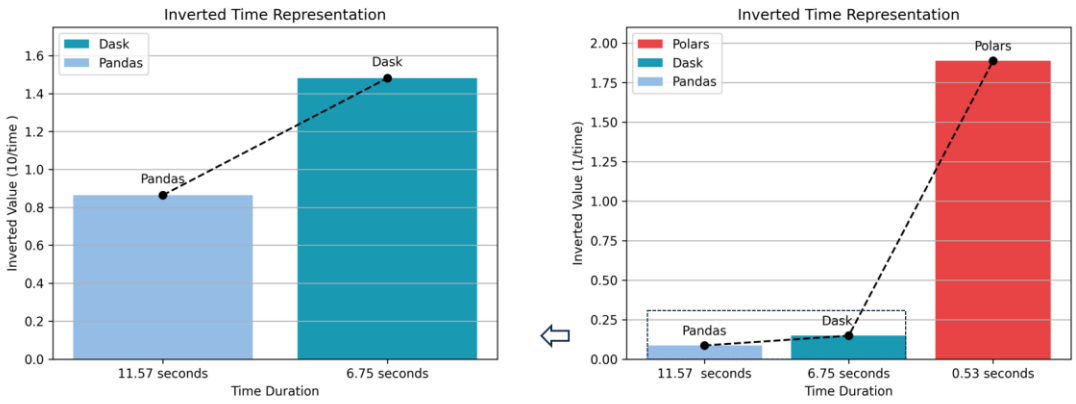

图 2 理论速度对比 1

Figure 2 Theoretical Speed Comparison 1

根据理论速度对比 1 显示，相较于 Pandas 库，Dask 库在运行速度方面已经实现了接近倍数的速度提升。而引入 Polars 库后，整体运行速度得到了显著提升，进一步验证了 Polars 在处理大规模数据时的卓越性能优势。

（3）实验内容 2：

使用循环结构的 Pandas、Dask 和 Polars 方法，以及多进程的 Polars 方法，分别对附件一中志愿者的加速度记录数据进行检索测试。针对数据列“annotation”中值为“7030 sleeping; MET 0.95”的记录，执行检索操作，并计算总消耗时间。

实验结果如图 3 所示：

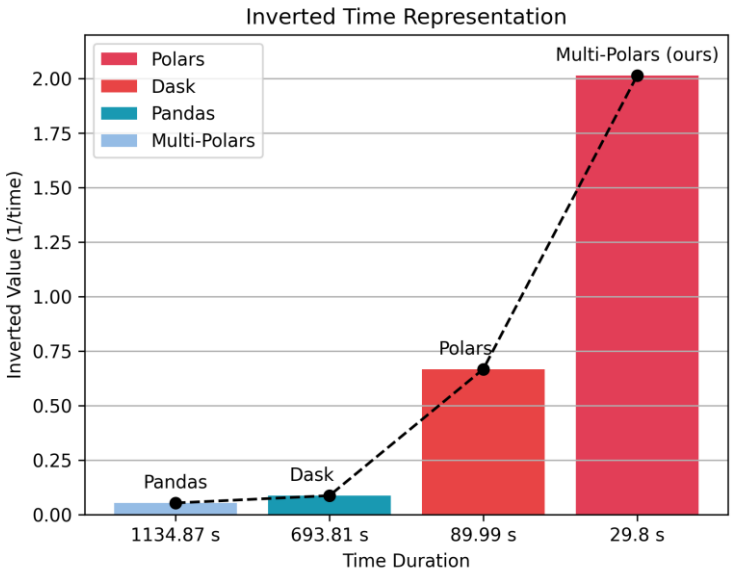

图 3 理论速度对比 2

Figure 3 Theoretical Speed Comparison 2

根据理论速度对比结果 2 显示，相较于 Pandas 方法，理论速度提升了 38 倍；而相较于顺序执行的 Polars 程序，经过多进程优化后，理论速度提升了近 3 倍。合理的多进程优化进一步缩短了计算时间，提升了整体处理能力。

实验结果总结：**1）在单文件处理方面，Polars 框架相比经典方法实现了约为 22 倍的性能提升；2）在多线程并行处理方面，相较于循环结构的 Polars 方法，性能提升约为 3 倍。**

2.4 探索性数据分析（EDA）

探索性数据分析（Exploratory Data Analysis，简称 EDA）是一种通过可视化和统计方法来分析数据集，揭示其潜在模式、特征和关系的过程。EDA 的主要目标是通过对数据的初步了解，为后续的建模和分析提供指导。

首先，使用基于 Polars 库编写的 Python 代码对数据进行了重复值和缺失值检测。检测结果显示，数据的时间间隔严格为 0.01 秒，频率为 100Hz，验证了数据采集的精度和一致性。数据中未发现重复值，进一步确保了数据的独立性和可靠性。

然而，发现志愿者编号为“P004、P049、P057、P073 和 P077”的数据存在时间戳“跳跃”的问题。

表 1 是 P004.csv 文件中时间戳“跳跃”的示例：

表 1 时间戳“跳跃”数据

Table 1 Timestamp “jump” data

time	x	y	z	annotation
2016-10-31 01:27:59.970000	0.9700076	-0.25264776	0.047496043	7030 sleeping;MET 0.95
2016-10-31 01:27:59.980000	0.9700076	-0.23707241	0.06330547	7030 sleeping;MET 0.95
2016-10-31 01:27:59.990000	0.9700076	-0.25264776	0.047496043	7030 sleeping;MET 0.95
2016-10-31 02:28:00.000000	0.9700076	-0.23707241	0.047496043	7030 sleeping;MET 0.95
2016-10-31 02:28:00.010000	0.9700076	-0.23707241	0.047496043	7030 sleeping;MET 0.95
2016-10-31 02:28:00.010000	0.9700076	-0.23707241	0.047496043	7030 sleeping;MET 0.95
2016-10-31 02:28:00.020000	0.9700076	-0.23707241	0.047496043	7030 sleeping;MET 0.95

加速度记录的时间从“2016-10-31 01:27:59.990000”直接跳跃至“2016-10-31 02:28:00.000000”。本文假设这一跳跃是由于数据记录问题导致的时间戳缺失，且该缺失的占比较小，无法进一步处理。同时，由于这一跳跃不会影响数据的时间间隔（频率），因此决定不对其进行修正。

接着，对“annotation”列中缺失时间区间进行了统计与异常值检测。异常值检测方法包括检查时间间隔是否超出正常的时间范围或是否出现极端偏差，从而识别出可能的数据录入错误或传感器故障。此过程有助于确保数据的准确性，并为后续的分析提供更加可靠的数据基础。

DBSCAN（Density-Based Spatial Clustering of Applications with Noise）是一种基于密度的空间聚类算法，能够识别复杂数据集中形状不规则的簇，并有效处理噪声。基于 DBSCAN 算法的异常检测是一种无监督学习方法，无需先验知识或标签即可识别数据中的异常值。

首先，对加速度计数据进行预处理，利用 Polars 库统计“annotation”列中缺失值的持续时间，并将这些持续时间构建成一个列表。接下来，使用 DBSCAN 算法对该列表数据进行异常检测。具体的算法步骤如下：

（1）构建持续时间列表：将每一段空值的持续时间统计并提取为一个列表，记为$T=[t_1,t_2,...,t_n]$。

（2）应用 DBSCAN 算法：DBSCAN 的基本思想是：如果一个点的密度足够大（即在其邻域内至少包含 *MinPts*（在领域的半径内所需的最小点数）个点），则该点属于一个簇；否则，它是噪声点。

通过以下公式来判定一个点是否属于簇：

核心点：对于数据点 p，若其 ϵ 领域内包含至少 *MinPts* 个点，则 p 是一个核心点。

$$CorePoint(p) \Leftrightarrow \left|\left\{q \middle| distance(p,q) \le \epsilon\right\}\right| \ge MinPts \tag{2}$$

（3）异常检测：使用 DBSCAN 算法对持续时间列表 T 进行聚类，旨在识别出潜在的异常值，即那些不属于任何簇的噪声点。

空值区间的时间统计结果如图 4 所示（以 P002.csv 为例）。表头包括缺失值的区间的开始时间、结束时间、缺失值前后的 MET 值，以及空值的持续时间。

Time1(Start)	Met1	Time2(End)	Met2	Time_diff(h)	Met_diff
2016-04-24 03:59:00.010000	0.95	2016-04-24 05:45:25	4	1.77361	3.05
2016-04-24 05:48:52.010000	4	2016-04-24 05:50:01.010000	8	0.0191667	4
2016-04-24 06:11:25.030000	8	2016-04-24 06:12:34.030000	4	0.0191667	4
2016-04-24 06:14:53.030000	4	2016-04-24 06:37:35.050000	2	0.378339	2
2016-04-24 06:42:11.050000	2	2016-04-24 06:43:21.050000	1.3	0.0194444	0.7
2016-04-24 06:46:45.060000	1.3	2016-04-24 06:47:54.060000	2	0.0191667	0.7
2016-04-24 06:52:31.060000	2	2016-04-24 06:53:41.060000	1.3	0.0194444	0.7
2016-04-24 06:58:22.070000	1.3	2016-04-24 06:59:32.070000	2	0.0194444	0.7
2016-04-24 07:01:50.070000	2	2016-04-24 07:02:59.070000	1.3	0.0191667	0.7
2016-04-24 07:30:30.090000	1.3	2016-04-24 07:31:38.090000	3.3	0.0188889	2
2016-04-24 07:40:52.100000	3.3	2016-04-24 07:42:02.100000	1.3	0.0194444	2
2016-04-24 07:46:55.110000	1.3	2016-04-24 07:48:03.110000	1.5	0.0188889	0.2
2016-04-24 07:58:18.120000	1.5	2016-04-24 07:59:25.120000	1.3	0.0186111	0.2

图 4 空值区间时间统计

Figure 4 Null interval time statistics

缺失值持续时间异常检测如图 5 所示：

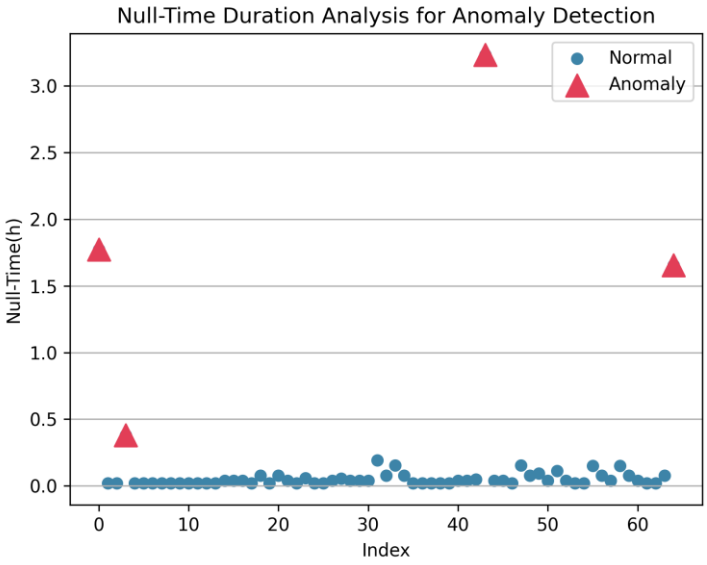

图 5 缺失值持续时间异常检测

Figure 5 Missing value duration anomaly detection

从图中结果可以看出，大部分缺失值的持续时间都集中在一个较短的范围内，通常与数据状态转变时的缓冲时间相关。

然而，仍然存在一些缺失值的持续时间达到数小时，形成较大的空值区间。因此，在后续分析中，需要关注这些长时间空值区域，并对其进行综合分析。

数据中的“annotation”列缺失值呈现出一定的规律性。具体而言，该列在一段时间内通常保持完整，即没有缺失数据，随后会出现一段持续的缺失值，之后又恢复为一段时间的完整值。这种模式可能与数据采集过程中的特定周期、系统状态变化或运动状态识别算法有关。

图 6 数据缺失值区间规律

Figure 6 Patterns of missing data value intervals

如图 6 所示，MET2（表示一段相同值的区间）和 MET4 为缺失值区间；其中，MET2 位于 MET1 与 MET3 之间，充当了相邻状态之间的“过渡点”。本文将其视为相邻两种状态之间的“转变区”。

以下通过统计进一步展示空值区间的规律，如图 7 所示：

start_index	end_index	annotation
i64	i64	str
0	1668000	7030 sleeping;MET 0.95
1668001	2209999	EMPTY
2210000	2255100	home activity;miscellaneous;si…
2255101	2282199	EMPTY
2282200	2289000	home activity;miscellaneous;wa…
2289001	2289099	EMPTY
2289100	2289400	home activity;miscellaneous;wa…
2289401	2299199	EMPTY
2299200	2325600	home activity;household chores…
2325601	2328099	EMPTY
2328100	2346500	home activity;miscellaneous;si…
2346501	2348999	EMPTY
2349000	2368700	home activity;miscellaneous;wa…
2368701	2371199	EMPTY
2371200	2409800	home activity;household chores…

图 7 区间统计

Figure 7 Interval statistics

在探索过程中发现，除 P052.csv 文件数据结尾为空外，其余数据文件中的缺失值区间均位于两个非空状态之间。基于这一观察，本文提出的数据填补方案是通过前后两个非空区间的 MET 值均值来填补缺失数据。

首先，需要构建正则表达式以提取“annotation”中的 MET 值。构建的正则表达式为“;MET (\d+.\d+)”。以“7030 sleeping; MET 0.95”为例，正则表达式提取出的 group1 内容为“0.95”。

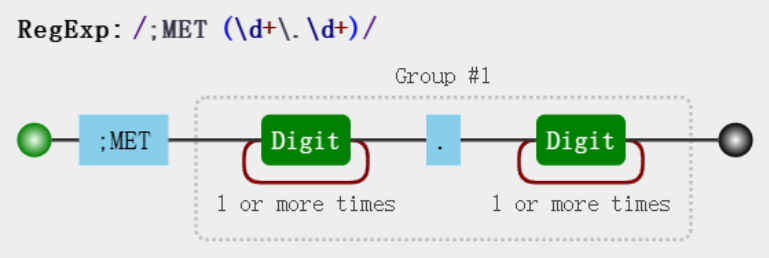

图 8 正则表达式可视化

Figure 8 Regular expression visualization

随后，采用向前填充与向后填充的平均值来填补空值区间数据，具体的插补公式如下：

$$x_i = \begin{cases} \dfrac{ForwardFill(x_i) + BackwardFill(x_i)}{2}, x_i = null \\ x_i, x_i \neq null \end{cases} \tag{3}$$

对于 P052.csv 文件数据结尾为空的情况，则采用向前填充进行数据填补。

数据插补如图 9 所示：

[1, 1, 1, None, None, None, 2, 2, 2, None, None, None, 3, 3, 3, None, None, None]

[1, 1, 1, 1.5, 1.5, 1.5, 2, 2, 2, 2.5, 2.5, 2.5, 3, 3, 3, 3, 3, 3]

图 9 数据插补示例

Figure 9 Data interpolation example

2.5 MET 值统计

在探索性数据分析中发现，数据的采样频率为标准的 100Hz，即每秒钟采集 100 个数据点。因此，在统计 MET 值的时间时，可以直接通过计算数据量并除以 360000（将其转换为小时），从而无需采用传统的时间差值计算方式。

```
data = (
    data
    .with_columns(
        pl.col('annotation').str.extract(r';MET(\d+\.\d+)',1).cast(pl.Float64).alias('
MET')
)
    .pipe(segment_mean_imputation, 'MET')
)

result = (
    data.group_by('MET')
    .agg(pl.col('time').count().alias('data_count'))
    .with_columns(
        (pl.col('data_count') / 360000).round(4).alias('estimated_time_seconds')
    )
)
```

问题一统计汇总流程如下：

（1）提取 MET 值：通过构建正则表达式“;MET(\d+\.\d+)”提取 annotation 列中的 MET 值。

（2）填补缺失值：使用 segment_mean_imputation 函数对 MET 列中的空值进行填补，填补方式是基于前后两个非空状态的均值。

（3）按 MET 分组并计算数据统计：使用 group_by('MET')将数据按 MET 值分组，并计算每个组中 time 列的有效数据数量。

（4）估算时间：计算每个 MET 组的有效数据所占的时间。将 data_count 除以 360000 即可（数据频率为 100Hz，每秒采样 100 次，每小时 360000 次采样）。

该流程的目标是提取、填补缺失值，并对数据进行汇总分析，最终得到每个 MET 状态下有效数据的数量和估算时间。

注：由于该问中仅涉及“annotation”列的统计，因此在数据读取时可以只加载“time”列和“annotation”列，从而缩短处理时间。

基于 Multi-Polars 模型对附件 1 中志愿者加速度记录数据进行统计汇总，单个文件（P001.csv）统计结果如图 10 所示：

MET --- f64	total_time_diff --- duration[μs]	total_time_diff_seconds --- f64	total_time_diff_hours --- f64	total_time_diff_hours_rounded --- f64
0.95	10h 35m 10ms	38100.01	10.583336	10.5833
1.225	3h 31s 980ms	10831.98	3.008883	3.0089
1.3	1h 33m 10s 80ms	5590.08	1.5528	1.5528
1.4	5m 56s 900ms	356.9	0.099139	0.0991
1.5	6h 36m 27s 260ms	23787.26	6.607572	6.6076
1.65	9m 6s 900ms	546.9	0.151917	0.1519
1.75	26m 23s 780ms	1583.78	0.439939	0.4399
1.8	22s 990ms	22.99	0.006386	0.0064
1.8	24m 3s 70ms	1443.07	0.400853	0.4009
1.9	1m 36s 960ms	96.96	0.026933	0.0269
2.0	1h 59s 190ms	3659.19	1.016442	1.0164
2.15	1m 10s 970ms	70.97	0.019714	0.0197
2.25	1m 36s 960ms	96.96	0.026933	0.0269
2.3	1h 16m 44s 20ms	4604.02	1.278894	1.2789
2.4	9m 33s 860ms	573.86	0.159406	0.1594
2.5	25m 15s 60ms	1515.06	0.42085	0.4209
2.65	3m 37s 940ms	217.94	0.060539	0.0605
2.9	22s 990ms	22.99	0.006386	0.0064
3.0	45m 25s	2725.0	0.756944	0.7569
3.15	1m 16s 980ms	76.98	0.021383	0.0214
3.25	23s 990ms	23.99	0.006664	0.0067
3.3	2h 40m 19s 90ms	9619.09	2.671969	2.672
4.0	20m 34s 20ms	1234.02	0.342783	0.3428

图 10 单个文件时间统计

Figure 10 Individual file time statistics

由于在数据转换过程中，可能会出现四舍五入引起的精度偏差。因此，为了确保结果的合理性，进行了平均时间差值的计算。

$$平均时间差=\frac{\sum_{i=1}^{n}\left|记录总时长_i-各活动时长总和_i\right|}{100} \tag{4}$$

最终得出的计算结果为 0.00011，该值在可接受的精度范围内。

Multi-Polars 运行结果如图 11 所示：

```
PID: P100
shape: (1, 6)
```

高强度总时长（小时）	中等强度总时长（小时）	低强度总时长（小时）	静态行为总时长（小时）	睡眠总时长（小时）	总时长（小时）
f64	f64	f64	f64	f64	f64
0.0964	6.1807	5.1997	7.6898	7.25	26.4167

```
Time cost: 51.31 seconds
```

图 11 Multi-Polars 运行结果

Figure 11 Multi-Polars run results

最终，整个统计汇总任务在 51.31 秒内完成。值得注意的是，整个处理流程的实现代码可精简至 100 行以内，充分展示了该方案的高效性和低成本。

三、基于 StatXBoost 的 MET 值估计模型

3.1 问题分析

问题：（1）对附件 2 中 20 位志愿者的性别、年龄信息及时间与加速度计数据进行 MET 值预测，并将每一位志愿者的 MET 值预测结果保存在 result_2 的文件夹中。（2）将 20 位志愿者的 MET 值预测结果进行整理，统计汇总各个志愿者的运动强度信息，数值保留小数点后 4 位，将结果整理至论文中，汇总格式见表 2，保存为“ result_2.xlsx”文件。

分析：基于在前问中识别出的加速度数据区间的规律，可以对各个区间的加速度数据进行统计特征和频域特征的提取。同时，将性别、年龄等信息转化为特征，以提升模型的泛化能力，进而构建一个新的特征数据集。基于该数据集，进行回归任务的建模，并保存训练好的模型，最终对附件 2 中 20 位志愿者的 MET 值进行预测。

因此，本问题需要解决的两个关键点：**1）如何构建加速度特征数据集；2）如何选择合适的模型进行建模并预测。**

3.2 Result_1 统计分析

将“Metadata1.csv”与“result_1.xlsx”文件进行合并，对其进行分析，观察其中年龄、性别等与活动时间的相关性，为后续建模打下基础。

pid	age	sex	记录总时长（小时）	睡眠总时长（小时）	高等强度运动总时长（小时）	中等强度运动总时长（小时）	低等强度运动总时长（小时）	静态活动总时长（小时）
str	str	str	f64	f64	f64	f64	f64	f64
P001	38-52	F	29.6667	10.5833	0.0	3.7998	4.0151	11.2684
P002	38-52	F	26.0833	6.25	0.395	2.0067	7.6837	9.748
P003	53+	M	27.1667	6.6667	0.0	7.0145	5.3504	8.1352
P004	38-52	F	27.0	9.9107	0.0	2.604	4.5181	9.9673
P005	53+	M	25.0	4.3333	0.0	4.0199	4.9264	11.7206
P006	53+	M	25.5	4.0833	0.0	2.5096	4.8708	14.0363
P007	53+	M	29.25	8.25	0.4209	0.4286	7.9129	12.2375
P008	30-37	M	28.45	6.6667	0.0594	2.3185	4.5236	14.8819
P009	18-29	F	27.0	6.0	0.0	4.5756	1.2609	15.1635
P010	30-37	F	26.5	9.3333	0.0	2.2082	7.3652	7.5933
P011	38-52	F	27.5	6.1667	0.0	0.295	12.433	8.6056
P012	30-37	F	26.0	6.6667	0.0	3.3911	11.0435	4.8988
P013	30-37	F	26.5	9.0	0.3539	0.743	4.3939	12.0092
P014	38-52	F	25.5	5.1667	0.0	1.3517	2.5702	16.4116
P015	53+	F	25.5	4.6667	0.0	0.7253	3.8651	16.2428

图 12 数据合并展示（部分）

Figure 12 Data consolidation display (partial)

表 2 活动时间统计

Table 2 Activity time statistics

年龄	平均记录总时长	平均睡眠总时长	平均高等强度运动	平均中等强度运动	平均低等强度运动	平均静态活动
18-29	26.5288	7.3947	**0.1008**	1.7941	6.0065	11.2328
30-37	26.6094	7.5308	0.0227	2.1973	6.677	10.1816
38-52	26.077	6.433	0.0483	1.8934	6.6946	11.0077
53+	26.5417	6.1013	0.0423	2.8983	5.6606	11.8392

根据 result_1 与 Metadata 中志愿者年龄数据的汇总结果显示，18-29 岁年龄段的志愿者在高强度运动中的平均 MET 值明显高于其他年龄组。这也表明，在构建 MET 预测模型时，除了加速度计数据外，考虑年龄、性别等因素对于提高模型的泛化能力至关重要。

分别使用饼图对年龄分布和性别分布进行可视化，如图 13 所示：

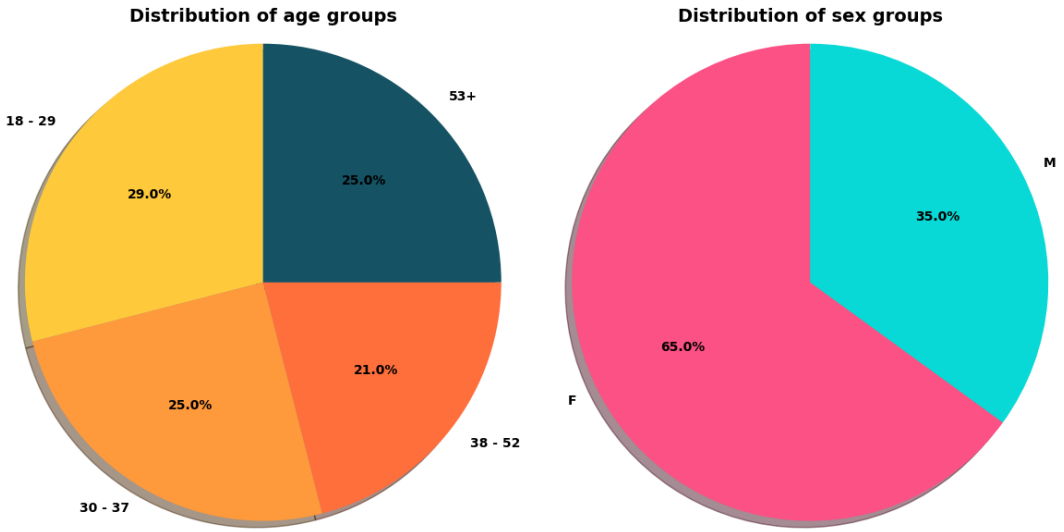

图 13 年龄分布与性别分布

Figure 13 Age distribution vs. gender distribution

从性别和年龄分布的饼图中可以看出，年龄分布较为均匀，各个年龄段的人群比例相对平衡；而在性别分布方面，男女比例接近于 1:2。

3.3 数据预处理

在问题一中，缺失值通过前后均值填补进行估计。而在问题二中，由于需要基于加速度记录数据进行 MET 值的预测，因此采用了不同的策略：删除缺失值，并仅对其余有效数据进行建模。

首先，对加速度数据进行预处理和信号滤波。数据预处理主要包括修正异常值，以确保数据的准确性和一致性。接着，通过信号滤波技术对加速度信号进行平滑处理，有效去除高频噪声和不必要的波动，从而提升后续分析的精度与可靠性。

孤立森林（Isolation Forest，简称 iForest）是一种用于异常检测的算法，特别适用于高维数据。其核心思想是通过构建一系列孤立树（Isolation Tree），将数据点“孤立”开来，并根据样本点被孤立的难易程度来判断其是否为异常点。相比其他异常检测方法，孤立森林具有较高的计算效率，尤其适用于大规模数据集。

孤立森林通过构建多个孤立树，并将每个数据点在所有树中的路径长度进行平均，最终计算出该点的异常分数（Anomaly Score）。异常分数较高的样本点被认为是异常点。异常分数的计算公式为：

$$S(x)=2^{-\frac{E(h(x))}{c(n)}} \tag{5}$$

其中，$S(x)$是样本点x的异常分数，$E(h(x))$是样本点x在孤立树中的平均路径长度，$c(n)$是一个常数，依赖于数据集的大小n。

滑动滤波（Sliding Window Filter）是一种通过在数据上滑动一个固定大小的窗口来处理数据的技术。它可以用来平滑信号或去除噪声，常用于时间序列分析和图像处理。

本文采用均值滤波对加速度数据进行处理，数学公式如下：

$$y_n = \frac{1}{w}\sum_{i=n-\frac{w}{2}}^{n+\frac{w}{2}} x_i \tag{6}$$

其中，w是窗口大小，x_i是数据点。

首先，使用孤立森林算法进行异常检测剔除异常值，随后应用滑动均值滤波（窗口大小为 10）对加速度数据进行平滑处理，以消除噪声并确保数据稳定性。

具体公式如下：

$$\begin{cases} \left\{x_1^{'}, x_2^{'}, ..., x_n^{'}\right\} = Isolation\left(\left\{x_1, x_2, ..., x_n\right\}\right) \\ y_n = \dfrac{1}{w}\displaystyle\sum_{i=n-\frac{w}{2}}^{n+\frac{w}{2}} x_i \\ y_n = \dfrac{1}{10}\left(x_{n-5}^{'} + x_{n-4}^{'} + ... + x_{n+4}^{'}\right) \end{cases} \tag{7}$$

实验结果如图 14 所示（以 x 轴为例）：

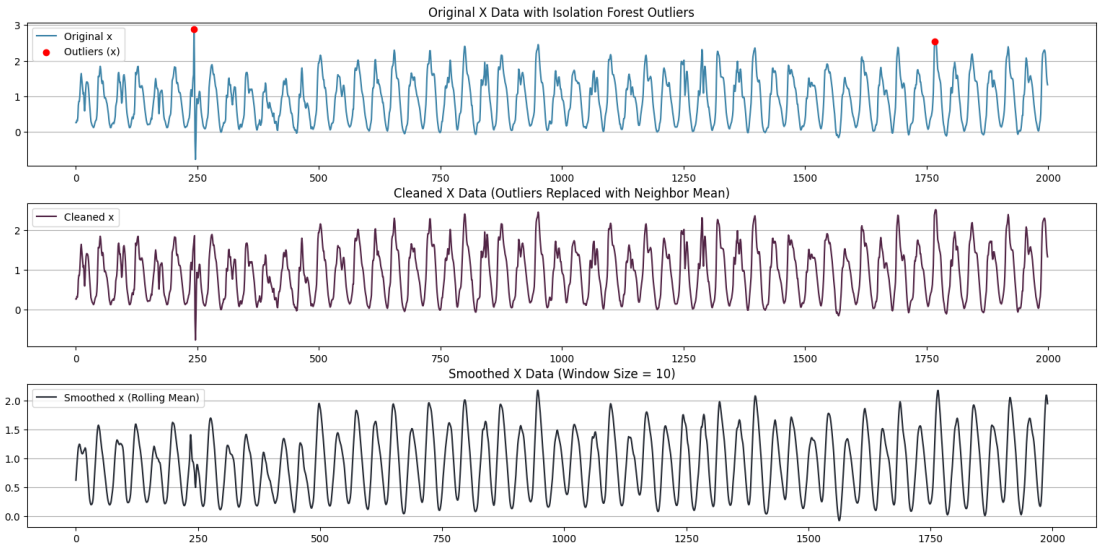

图 14 数据异常处理与平滑滤波

Figure 14 Data anomaly handling and smoothing filtering

3.4 特征工程

特征工程是数据预处理过程中的关键环节，旨在从原始数据中提取有意义的特征，从而提升模型的预测能力和性能。它包括特征选择、特征构造、特征转换和特征缩放等步骤。结合前文分析，对于加速度数据，选择通过构建各个区间的统计特征和频域特征来进行特征工程，以充分挖掘数据中的潜在信息。

统计特征是从信号的时间域数据中提取的基本统计量，用于描述信号的分布、变化趋势和波动情况。基于加速度数据构造的统计特征如下：

（1）均值

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{8}$$

其中 x_i 是信号在第 i 个时间点的值。

（2）标准差

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i - \mu_x\right)^2} \tag{9}$$

其中 μ_x 是信号的均值， x_i 是信号在第 i 个时间点的值。

频域特征是通过傅里叶变换（FFT）将信号从时间域转换到频域后，提取的一些信息。频域分析有助于理解信号的频率成分及其分布特征。常见的频域特征包括：

（1）最大频率

$$f_{\max} = \arg\max\left(\left|FFT\left(x\right)\right|\right) \tag{10}$$

其中 x 是信号，$FFT\left(x\right)$ 是信号的傅里叶变化结果，$f_{\max}$ 是振幅最大的频率。

（2）频谱振幅

$$SpecAmp = \sum_{i=0}^{N/2}\left|FFT\left(x\right)\left[i\right]\right| \tag{11}$$

其中 N 是信号的长度， $FFT\left(x\right)\left[i\right]$ 是信号的第 i 个 FFT 系数。

（3）频率中心

$$f_{center} = \frac{\sum_{i=0}^{N/2} f_i \cdot \left|FFT\left(x\right)\left[i\right]\right|}{\sum_{i=0}^{N/2}\left|FFT\left(x\right)\left[i\right]\right|} \tag{12}$$

其中 f_i 是第 i 个频率， $\left|FFT\left(x\right)\left[i\right]\right|$ 是该频率的振幅。

相关性特征通过计算不同信号轴（x、y、z）之间的相关系数来提取，以揭示各方向加速度之间的关系和相互影响。

皮尔逊相关系数（Pearson Correlation Coefficient）是衡量两个变量之间线性关系强度和方向的统计指标，取值范围从-1 到 1，其计算公式为：

$$r=\frac{\sum\left(X_i-\bar{X}\right)\left(Y_i-\bar{Y}\right)}{\sqrt{\sum\left(X_i-\bar{X}\right)^2\sum\left(Y_i-\bar{Y}\right)^2}} \tag{13}$$

其中，X_i 和 Y_i 分别是两个变量的观测值，$\bar{X}$ 和 $\bar{Y}$ 是它们的均值。

$$\begin{cases} corr\left(x,y\right)=\dfrac{\mathrm{cov}\left(x,y\right)}{\sigma_x\sigma_y} \\ corr\left(x,z\right)=\dfrac{\mathrm{cov}\left(x,z\right)}{\sigma_x\sigma_z} \\ corr\left(y,z\right)=\dfrac{\mathrm{cov}\left(y,z\right)}{\sigma_y\sigma_z} \end{cases} \tag{14}$$

其中 $\mathrm{cov}\left(x,y\right)$ 是 x 和 y 的协方差，σ_x 和 σ_y 是它们的标准差，以此类推。

其他特征需要对年龄和性别进行数据转换。根据第 4.2 节的分析，年龄被划分为四个区间：“18-29 岁”、“30-37 岁”、“38-52 岁”和“53 岁及以上”。性别则分为“F”（女性）和“M”（男性）。因此，需要对这两个特征进行相应的转换，转换规则如表 3 和表 4 所示：

表 3 年龄数据转换

Table 3 Age data conversion

年龄区间	转换后
18-29	24
30-37	34
38-52	45
53+	53

表 4 性别数据转换

Table 4 Gender data conversion

性别	转换后
M	1
F	0

所有构建的特征如表 5 所示：

表 5 特征集

Table 5 Feature set

特征	公式
均值	$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$
方差	$\sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)^2}$
最大频率	$f_{\max} = \arg\max\left(\left\|FFT(x)\right\|\right)$
频谱振幅	$SpecAmp = \sum_{i=0}^{N/2}\left\|FFT(x)[i]\right\|$
频率中心	$f_{center} = \frac{\sum_{i=0}^{N/2} f_i \cdot \left\|FFT(x)[i]\right\|}{\sum_{i=0}^{N/2}\left\|FFT(x)[i]\right\|}$
皮尔逊相关系数	$corr(x,y) = \frac{\mathrm{cov}(x,y)}{\sigma_x \sigma_y}$
年龄	-
性别	-

特征构造运行过程如图 15 所示：

shape: (1, 18)

x_mean	x_std	y_mean	y_std	z_mean	z_std	x_spectral_mean_amplitude	x_freq_centr	y_spectral_mean_amplitude	y_freq_centr	z_spectral_mean_amplitude	z_freq_centr	corr_xy	corr_xz	corr_yz	age	sex	annotation
f64	f64	f64	f64	f64	f64	f64	f64	f64	f64	f64	f64	f64	f64	f64	i32	i32	str
0.3655	0.4656	-0.1794	0.1888	-0.6765	0.345	3.2583e7	12.2948	2.6439e7	14.5073	2.4069e7	11.109	0.3626	0.7616	0.3364	45	0	7030 sleeping; MET 0.95

图 15 特征构造示例

Figure 15 Example of feature construction

3.5 StatXBoost 模型

StatXBoost 模型融合了特征构建中的统计方法与 XGBoost 模型的优势。在特征构建过程中，StatXBoost 引入了统计学方法，如均值、方差、相关性以及频域特征等，从而能够更有效地挖掘数据中的潜在规律和重要特征。同时，利用 XGBoost 这一强大的梯度提升算法进行模型训练和优化，进一步提升了预测精度和模型的泛化能力。

（1）XGBoost 模型

XGBoost（Extreme Gradient Boosting）是一种高效的梯度提升树（GBDT）实现，广泛应用于机器学习任务中，尤其是在处理结构化数据时，如分类、回归和排序问题。XGBoost 基于梯度提升算法（Gradient Boosting），通过结合多个弱分类器（通常是决策树）来提升模型的准确度，每棵树都在前一棵树的基础上进行改进，从而减少模型的误差。

XGBoost 的一个显著特点是其高效的计算性能。它通过优化算法，如分布式计算、数据并行化和硬件加速等，显著提高了模型的训练速度。该模型支持多线程，能够充分利用多核处理器，显著加快训练过程。此外，XGBoost 还引入了 L1（Lasso）和 L2（Ridge）正则化，有效防止模型的过拟合，并能在面对复杂数据时，控制模型复杂度，提高其泛化能力。

运用 XGBoost 模型进行 MET 值估计模型数学公式如下：

1.基本的 XGBoost 模型公式

对于加速度数据的统计特征与频域特征，构建样本 $x=\left(x_1,x_2,...,x_d\right)$（其中 d 是特征的维度），XGBoost 模型通过多个树的集成来计算最终的预测值。每棵树 t 的预测值是一个加法模型，其数学表达式为：

$$f_t\left(x\right)=\sum_{i=1}^{T}\hat{y}_i\cdot I\left(x\in R_i\right) \tag{15}$$

其中，T 是树的数量，$\hat{y}_i$ 是第 i 棵树的输出值，$I\left(x\in R_i\right)$ 是指示函数，表示样本 x 是否落在树 t 中的叶节点 R_i 上。

2.XGBoost 模型的加法形式

XGBoost 模型的预测过程是多个决策树的加法形式。假设有 M 棵树，最终的预测值 $\hat{y}$ 由每棵树的预测结果的加权和构成：

$$\hat{y}\left(x\right)=\sum_{t=1}^{M}f_t\left(x\right)=\sum_{t=1}^{M}\sum_{i=1}^{T}\hat{y}_i\cdot I\left(x\in R_i\right) \tag{16}$$

3.梯度提升的优化过程

XGBoost 通过梯度提升算法来优化模型。每次新增一棵树，都会基于当前模

型的残差进行拟合。假设当前模型的预测为 $\hat{y}^{(k)}(x)$，在第 k 步时，XGBoost 通过计算残差 $r_i^{(k)}$ 来更新预测：

$$r_i^{(k)} = y_i - \hat{y}^{(k-1)}(x_i) \tag{17}$$

其中 y_i 是第 i 个样本的真实 MET 值。

4.损失函数和正则化

XGBoost 的优化目标不仅考虑模型的误差（如均方误差、对数损失等），还考虑正则化项来防止过拟合。XGBoost 的损失函数可以表示为：

$$L(\theta) = \sum_{i=1}^{N} \Upsilon(y_i, \hat{y}_i) + \sum_{t=1}^{M} \Omega(f_t) \tag{18}$$

其中，$\Upsilon(y_i, \hat{y}_i)$ 是损失函数，$\Omega(f_t)$ 是树的正则化项，通常包含树的复杂度（叶子节点数和分裂点数等），其形式为：

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{k=1}^{T} \omega_k^2 \tag{19}$$

其中，T 是树的叶子节点数，γ 和 λ 是正则化参数，ω_k 是叶节点的权重。

（2）Random Forest 模型

随机森林是一种基于集成学习（Ensemble Learning）的机器学习算法，属于 Bagging 类型的集成方法。它通过构建多个决策树（Decision Tree）并将它们的预测结果进行集成，从而提高模型的准确性和鲁棒性。随机森林广泛应用于分类、回归以及特征选择等任务。

工作原理：1）训练集采样：从原始数据集中通过自助法（bootstrap sampling）随机抽取多个子集。每个子集用于训练一棵决策树。2）特征随机选择：在构建每棵树时，不是考虑所有的特征，而是随机选择一部分特征用于节点分裂，这样可以增加树之间的多样性，减少过拟合。3）集成预测：对于回归任务，随机森林的预测值是各个决策树预测值的均值。

运用随机森林进行 MET 值估计模型数学公式如下：

数据集 $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$，其中 x_i 是输入特征，包括加速度数据的统计特征与频域特征，y_i 是目标变量即 MET 值，随机森林的回归预测可以表示为：

1.训练多颗决策树：

对于每颗决策树 T_k，它的预测值为 $\hat{y}_k(x)$，其中 x 是新的输入数据。

2.集成预测：

随机森林的最终预测值是所有树预测值的平均值：

$$\hat{y}_{RF}(x)=\frac{1}{K}\sum_{k=1}^{K}\hat{y}_k(x) \tag{20}$$

其中：

K 是树的总数（即森林的大小）。

$\hat{y}_k(x)$ 是第 k 棵树在输入 x 上的预测值。

随机森林是一种强大的集成学习方法，主要优点包括：能够处理高维数据，适用于分类和回归任务；通过构建多个决策树减少过拟合，提高模型的准确性；对缺失值有较好的鲁棒性，并能自动评估特征的重要性。它适用于大规模数据集，计算效率较高。

（3）LightGBM 模型

LightGBM (Light Gradient Boosting Machine) 是一个由微软开发的高效、灵活的梯度提升框架，用于构建机器学习模型，特别是在处理大规模数据时表现突出。LightGBM 是基于梯度提升决策树（GBDT，Gradient Boosting Decision Tree）算法构建的，但相对于传统的 GBDT，它在多个方面进行了优化，显著提高了计算效率和内存使用。

LightGBM 的核心是基于梯度提升树（GBDT）算法的，该算法的目标是通过一系列决策树的组合来最小化损失函数。

损失函数优化：通过梯度下降法，GBDT 的目标是最小化损失函数：

$$\hat{f}(x)=\arg\min\sum_{i=1}^{n}L\left(y_i,\ f(x_i)\right)+\Omega(f) \tag{21}$$

其中 L 为损失函数，$\Omega(f)$ 是对模型复杂度的正则化项。

随机搜索（Random Search）是一种常用于超参数优化的技术，它通过随机选择超参数空间中的不同组合来训练模型，寻找最佳的超参数组合。与网格搜索（Grid Search）相比，随机搜索不需要穷举所有可能的超参数值，而是从预定义的超参数空间中随机选择若干组合进行尝试。这种方法可以在计算资源有限的情况下，迅速找到接近最优的超参数配置。

随机搜索的优点之一是其相对较低的计算开销。网格搜索需要测试每一种可能的超参数组合，而随机搜索则只需选择一个随机的子集，这使得它在较高维度的超参数空间中表现得更加高效。此外，随机搜索在多维超参数空间中能够更灵活地探索不同的组合，特别是在有些超参数对模型性能影响较大时，它能够较为高效地找到好的超参数配置。

采用基于随机搜索的参数优化方法，分别对 3.4 节中构建的数据集（Features.csv）应用 XGBoost 模型、Random Forest 模型和 LightGBM 模型进行建模与预测。

首先需要对特征数据进行数据标准化，公式如下：

$$x_i^{'} = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{22}$$

x_i 是原始数据中的第 i 个数据点，$\min(x)$ 是数据中的最小值，$\max(x)$ 是数据集中的最大值，$x_i^{'}$ 是标准化的第 i 个数据点。

在对数据进行标准化处理后，接下来需要对数据集进行划分，将其按 8:2 的比例分为训练集和测试集。

选用 MAPE 与 MSE 作为评价指标，由于数据量的原因，模型并不能很好的反应出数据的变化，故不适用选用 R^2 作为评价指标。

MAPE 计算的是预测值与实际值之间的相对误差的平均百分比，用来评估预测模型的准确性，数学公式如下：

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{23}$$

MSE 衡量的是预测值与实际值之间的平方差的平均值，值越小表示模型预测越准确，数学公式如下：

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{24}$$

其中，y_i 是实际值（真实值），$\hat{y}_i$ 是预测值，n 是样本数量。

运行结果如表 6 所示

表 6 模型运行结果对比

Table 6 Comparison of model run results

评价指标/模型	XGBoost	RandomForest	LightGBM
MAPE	0.2895	0.2914	0.2913
MSE	0.6032	0.6130	0.6128

从上述评价指标可以看出，XGBoost、Random Forest 和 LightGBM 模型在预测任务中的表现差异较小。具体来说，三种模型的 MAPE 和 MSE 值非常接近，分别为 0.2895、0.2914 和 0.2913（MAPE），以及 0.6032、0.6130 和 0.6128（MSE）。这表明，尽管不同的模型和算法有各自的特点与优势，但它们在此数据集上的效果已经非常接近。

这一结果进一步强调了数据和特征的重要性。无论采用哪种机器学习算法，最终模型的性能主要还是受到数据质量和特征选择的限制。**机器学习模型和算法在本质上是通过不断优化来逼近数据集的潜力上限，而数据本身的特性和结构决定了这一上限。**因此，提升数据质量和合理的特征工程往往比单纯选择不同的算法更为重要。

基于 XGBoost 模型对数据进行建模，并绘制了预测对比图与残差图，结果如图 16 所示：

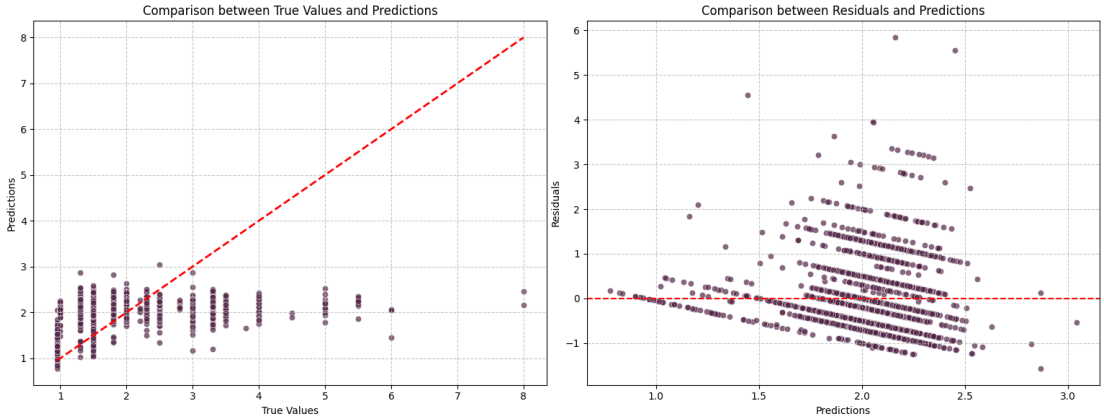

图 16 预测对比图与残差图

Figure 16 Prediction Comparison Plot vs. Residual Plot

通过预测对比图与残差图可以看出，模型在对 MET 值大于 3 的数据进行预测时表现较差。这主要体现在该部分数据的占比相对较小，导致模型在训练过程中未能充分学习到这类数据的特征。因此，当遇到这类数据时，模型的预测误差较大，无法做到准确预测。

3.6 睡眠匹配算法

在前文中，不仅发现了志愿者加速度数据中的区间规律，还注意到“annotation”列中包含的“7030 sleeping; MET 0.95”数据仅出现在数据的开头和结尾。根据定义，MET 值小于 1 被视为睡眠状态，因此，对于整个数据集而言，开头和结尾的数据主要反映并占据了睡眠状态。然而，3.5 节中的 StatXBoost 模型在预测附件 2 中志愿者的睡眠数据（即开头和结尾部分）时效果并不理想。为了改进这一问题，采用了一种次优方案：通过匹配附件 2 中志愿者的年龄、性别、开始记录时间和结束记录时间，结合附件 1 中对应志愿者的相关数据，将附件 1 中的“睡眠数据”用于填充附件 2 中的睡眠数据。这样可以有效弥补数据预测的影响。睡眠匹配算法的具体过程如下（以志愿者 P112 为例）：

（1）根据 P112 中的性别和年龄信息，对附件 1 进行检索并匹配，得到以下结果：

$$\{\text{P001},\text{P002},\text{P004},\text{P011},\text{P014},\text{P025},\text{P026},\text{P032},\text{P034},\text{P038}...\} \tag{25}$$

（2）计算时间差

$$\begin{cases} T_1 = \left| t_1^s - t_2^s \right| \\ T_2 = \left| t_1^e - t_2^e \right| \\ T = T_1 + T_2 \end{cases} \tag{26}$$

其中，T_1是开始时间之差，T_2是结束时间之差，以 P112 和 P001 为例（时间转换为分钟，不考虑年月日）。

$$\begin{cases} T_1 = \left|00{:}06{:}00.000000 - 00{:}05{:}00.000000\right| \\ T_2 = \left|02{:}11{:}00.000000 - 05{:}45{:}00.000000\right| \\ T = T_1 + T_2 = 215 \end{cases} \tag{27}$$

（3）选取最小时间差

$$MinT = \min\left(T^1, T^2, T^3 ...\right) \tag{28}$$

通过计算时间差，找出最小的时间差及其对应的志愿者编号。以 P112 为例，最小时间差出现在与 P002 之间，时间差为 5 分钟。

因为两者在性别、年龄和活动周期上高度相似，且开始和结束时间差异仅为 5 分钟。这样的相似性使得 P002 的睡眠数据成为 P112 的合理替代，可以有效填补睡眠数据，为后续分析打下基础。

总睡眠匹配表如表 7 所示：

表 7 睡眠时间匹配

Table 7 Sleep time matching

志愿者	匹配志愿者	时间差（分钟）
P101	P031	30
P102	P096	60
P103	P042	130
P104	P046	31
P105	P068	40
P106	P082	104
P107	P010	23
P108	P010	36
P109	P001	100
P110	P060	346
P111	P014	150
P112	P002	4
P113	P037	42
P114	P074	31
P115	P026	30
P116	P039	30
P117	P098	8
P118	P077	141
P119	P066	20
P120	P062	170

3.7 MET 值预测

在对 XGBoost 模型进行训练并保存模型文件后，以及对睡眠数据进行补充后，继续对附件 2 中 20 位志愿者的加速度数据进行预测。由于需要进行窗口区间的划分，基于附件 1 中 100 位志愿者数据的窗口划分结果，对其可视化如图 17 所示：

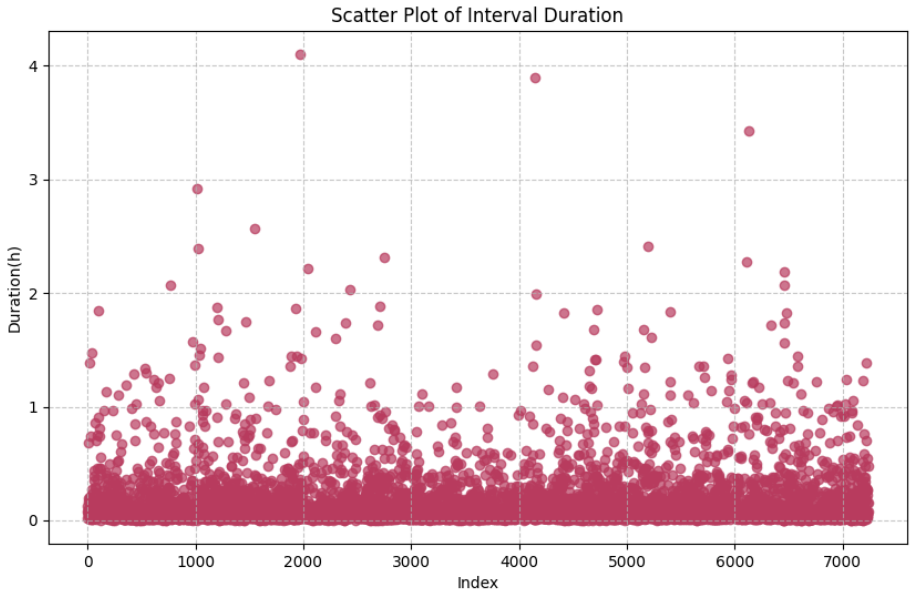

图 17 数据窗口散点图

Figure 17 Data Window Scatterplot

数据窗口的均值为 0.1453 小时，约为 52308 个数据点。本文选择以每 10000 个数据点作为步长进行预测，基于每个窗口中的 10000 个数据点构建统计特征和频域特征，并将这些特征作为输入提供给模型，最终预测出相应的 MET 值。

在进行数据预测后，基于问题一中的模型进行调整，并对运动活动信息进行统计汇总，结果如表 8 所示。

表 8 志愿者活动信息表（result_2）

Table 8 Volunteer activity information sheet (result_2)

志愿者 ID	记录总时长（小时）	高等强度运动总时长（小时）	中等强度运动总时长（小时）	低等强度运动总时长（小时	静态活动总时长（小时）
P101	25.5	0	1.0278	8.8125	10.8975
P102	24.5	0	0.1946	9.2066	9.2851
P103	29.1667	0	1.1667	11.5357	8.3675

志愿者 ID	记录总时长（小时）	高等强度运动总时长（小时）	中等强度运动总时长（小时）	低等强度运动总时长（小时	静态活动总时长（小时）
P104	26.3333	0	2.2792	8.5061	8.9793
P105	27	0	0	4.0865	14.3724
P106	28	0	0.7778	7.2445	13.0935
P107	26.25	0	0	9.7855	6.9499
P108	26.5	0	0.9167	9.2848	6.8109
P109	28	0	2.6944	5.6156	9.035
P110	27	0	1.1667	6.4215	12.9265
P111	27.8333	0	2.0556	5.4477	14.5386
P112	26.0833	0	1.9444	7.9508	9.7576
P113	26	0	1.4167	8.3946	12.3432
P114	24.5	0	3.4723	5.5305	10.1748
P115	24.5	0	1.2222	7.5336	11.2589
P116	26.5	0	2.25	7.8114	12.1597
P117	24.5	0	1.1667	8.1452	9.8411
P118	28	0.1667	4.6667	9.5784	8.4787
P119	25.5	0	1.25	6.31	13.2323
P120	26.3333	0	1.5278	6.9496	11.3144

在对志愿者预测数据进行统计汇总后，整体分布呈现出与问题一中志愿者统计汇总数据相似的趋势。具体而言，大多数预测结果与之前的数据较为一致，符合预期。然而，在高强度运动和中等强度运动（MET 值介于 3 到 6 之间）方面，预测结果普遍偏低。这意味着模型在这类活动的预测上存在一定的误差，未能充分捕捉到这些活动的运动强度和能量消耗特征。

这种现象与第 3.5 节中对模型在处理 MET > 3 数据时预测效果偏低的描述相呼应。进一步分析发现，模型可能未能准确地识别高强度活动的特征，尤其是在加速度数据的高频波动部分，导致对这些运动强度的 MET 值预测不足。

此外，模型在处理不同类型运动时可能存在对频域特征提取的不足，未能全面反映运动强度的变化。

四、基于差分与自适应阈值的睡眠阶段识别模型

4.1 问题分析

问题：随着健康意识的提高，人们越来越关注睡眠质量。加速度计作为一种非侵入式的可穿戴设备，为监测睡眠活动提供了一种便捷的方法。本题需设计并实现一个算法，该算法能够根据个体的加速度计数据，准确识别和分析个体的睡眠阶段及睡眠状态，为改善睡眠质量提供科学依据。将设计的方法或算法应用于附件 2 中的 20 位志愿者数据，给出具体识别结果，数值保留小数点后 4 位，并将结果整理至论文中，汇总格式见表 3，保存为“ result_3.xlsx” 文件。

分析：根据前文对加速度数据规律的分析，睡眠时间区间可以分为两个阶段，即开头阶段和结尾阶段。因此，需分别对这两段进行睡眠状态的划分。在观察睡眠阶段的数据时，发现存在一些突变点，这些突变点通常伴随着加速度数据的整体变化。因此，本文假设这些突变点为睡眠阶段的转换点，并将该问题简化为突变点检测任务。因此，本问题需要解决的关键点为**如何确定睡眠阶段突变点。**

4.2 差分与自适应阈值

由于第 3.6 节中的睡眠匹配算法是基于附件 1 中部分志愿者的睡眠数据进行匹配，因此本文选择对附件 1 中的加速度数据进行实验。以志愿者 P096 为例，首先可视化其睡眠时间内 MET 值的变化，结果如图 18 所示：

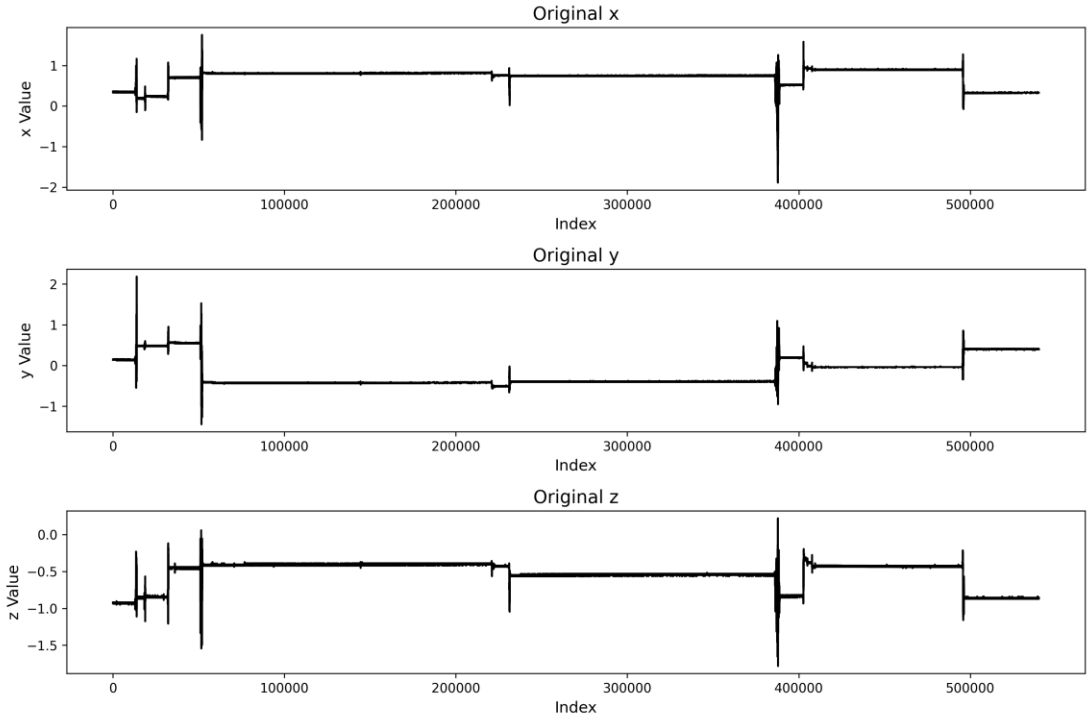

图 18 睡眠阶段 MET 值变化

Figure 18 Changes in MET values during sleep stages

从图中可以看出，x、y、z 三列数据呈现出相似的变化趋势，即在某个 MET 值发生显著变化后，数据会稳定到另一个值。基于这一观察，本文假设这些变化点为睡眠阶段的转换点。为了更清晰地突出这一变化，本文采用差分方法对数据进行处理，并以 x 方向的加速度数据为例进行分析。

差分（Differencing）是时间序列分析中的常用技术，主要用于去除数据中的趋势成分，使数据更加平稳，从而便于后续的分析和建模。差分的基本原理是通过计算相邻数据点之间的差值，消除时间序列中的长期趋势。

对 x 方向加速度进行差分并与原数据进行对比可视化如图 19 所示：

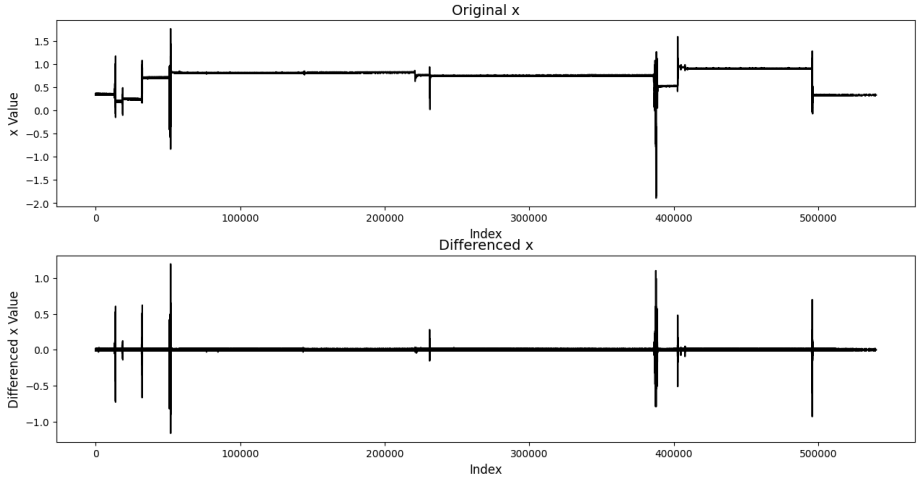

图 19 加速度差分可视化

Figure 19 Differential visualisation of acceleration

从图 19 可以看出，经过差分处理后，MET 值的变化变得更加明显。本文推测，MET 值发生较大变化的点可能对应睡眠阶段的转变点。因此，需设定一个阈值，以便准确识别这些转变点。

自适应阈值算法能够根据数据的变化自动调整阈值，从而更精确地识别数据中的重要转变点，例如睡眠阶段的变化。这种方法具有较高的灵活性，适用于各种变化模式的数据集。

自适应阈值算法流程如下：

（1）差分计算：

$$\Delta x(t) = x(t) - x(t-1) \tag{29}$$

（2）统计特征计算：

$$\mu_{\Delta x} = \frac{1}{N}\sum_{t=1}^{N} \Delta x(t) \tag{30}$$

$$\sigma_{\Delta x}=\sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(\Delta x(t)-\mu_{\Delta x}\right)^{2}} \tag{31}$$

（3）初始阈值设定：

$$\theta(t)=\mu_{\Delta x}+k\cdot\sigma_{\Delta x} \tag{32}$$

（4）动态阈值调整：

$$\theta(t)=\theta_{t-1}+\lambda\cdot\max\left|\Delta x(i)\right|, i\in[1,t] \tag{33}$$

（5）转变点检测：

$$\text{Point } t \quad \text{if} \quad |\Delta x(t)|>\theta(t) \tag{34}$$

自适应阈值算法的核心在于根据数据的差分变化和统计特征动态调整阈值，从而识别数据中显著变化的时刻，并判断是否为睡眠阶段的转变点。通过设定初始阈值并进行后续动态调整，该算法能够更精确地适应不同模式的数据变化，有效识别转变点。

自适应阈值算法睡眠阶段划分可视化如图 20 所示：

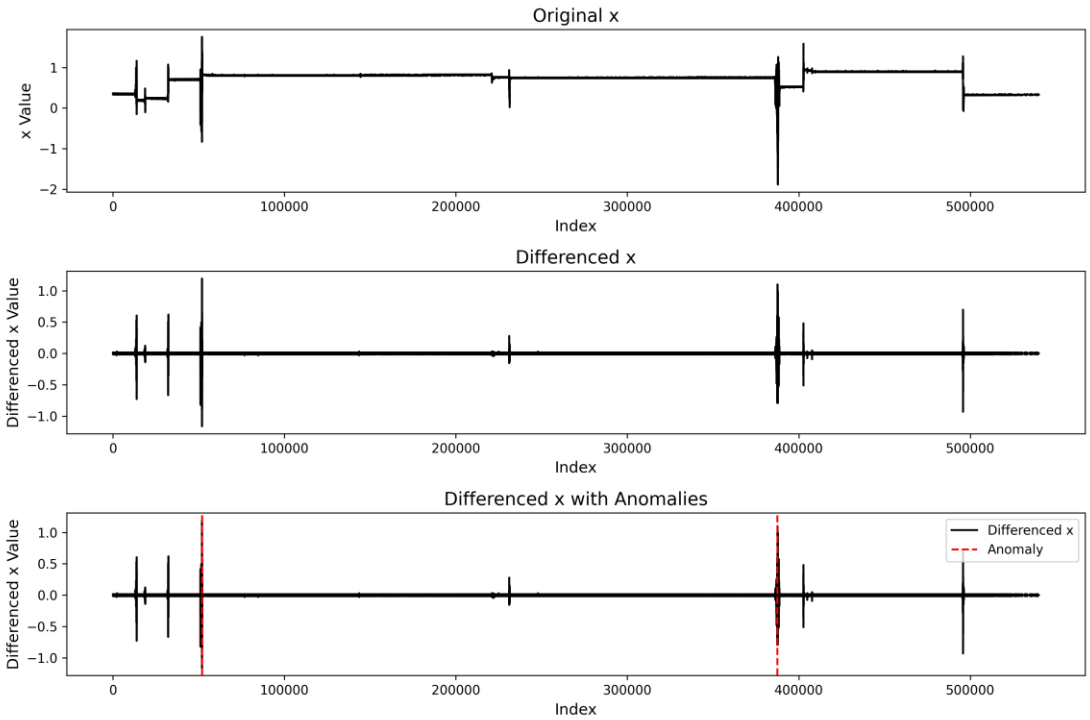

图 20 自适应阈值划分

Figure 20 Adaptive thresholding

如图 20 所示，通过自适应算法识别出两个睡眠状态的转变点，将整个睡眠过程划分为三个阶段。根据具体的睡眠时间，这些阶段可以进一步细分为入睡期、深睡期与眼动期。

4.3 睡眠阶段识别

将睡眠阶段检测模型应用到附件 2 中，识别结果如表 9 所示：

表 9 睡眠阶段识别表（result_3）

Table 9 Sleep stage recognition table (result_3)

志愿者ID	睡眠总时长（小时）	睡眠模式一总时长（小时）	睡眠模式二总时长（小时）	睡眠模式三总时长（小时）	睡眠模式四总时长（小时）	睡眠模式五总时长（小时）	睡眠模式六总时长（小时）	睡眠模式七总时长（小时）
P101	4.6667	1.2077	0.4696	0.5305	0.9588	4.6667	1.1329	0.3672
P102	5.8	0.6956	2.6264	0.9779	0.1444	0.9322	0.4235	
P103	7.8333	1.6475	3.1047	1.0812	0.0859	1.9141		
P104	6.5833	1.7931	1.7666	1.5238	0.6502	0.8496		
P105	8.1667	1.1311	1.4456	1.2967	0.7933	0.5858	2.9142	
P106	6.9	0.6024	0.8895	1.0081	1.5604	2.8396		
P107	9.3333	1.8776	1.091	0.8116	0.5532	2.7764	2.2235	
P108	9.3333	1.8776	1.091	0.8116	0.5532	2.7764	2.2235	
P109	10.5833	2.0031	2.6303	2.3884	3.5612			
P110	6.5	1.5364	1.4637	1.1821	1.3111	1.0067		
P111	5.1667	0.8732	0.8986	1.8949	1.2766	0.2234		
P112	6.25	1.8667	2.0499	0.8517	1.4817			
P113	3.8333	0.8589	1.2372	0.2472	1.393	0.107		
P114	5.3349	3.1678	1.3322	0.2822	0.5527			
P115	4.5	2.0288	0.4712	0.2986	1.1813	0.5201		
P116	4.2667	1.5898	1.1769	0.747	0.753			
P117	4.5	0.8932	1.6183	0.3219	0.2892	1.3774		
P118	4.65	0.4968	2.2794	0.3739	0.7059	0.794		
P119	4.6667	1.6912	1.4755	4.6667	1.0874	0.4126		
P120	6.0	1.3321	0.7169	1.0005	1.4506	0.3245	0.6808	0.4946

五、基于滑动窗口的久坐行为健康预警模型

5.1 问题分析

问题：久坐指连续处于静态坐姿单次超过 30 分钟，且以 MET 值 ＜1.6 为特征的行为。长时间久坐会增加多种慢性疾病的风险，导致多种健康问题。基于附件 1 文件夹中 100 位志愿者的加速度计数据及对应的活动行为，自动识别久坐行为状态，并在适当的时候发出预警，帮助用户了解和改善自己的活动习惯。将设计的方法或算法应用于附件 2 中的 20 位志愿者数据，并给出具体识别结果。

分析：久坐行为被定义为连续 30 分钟内 MET 值小于 1.6 的状态。根据第 3.7 节中预测时使用的窗口大小为 10000 个数据点，而 30 分钟对应的数据量为 180000 个数据点，即 18 个窗口。因此，可以采用滑动窗口算法，对连续 18 个 MET 区间进行检测，从而识别久坐行为的发生。

5.2 滑动窗口算法

滑动窗口算法（Sliding Window Algorithm）是一种在处理数据时，通过使用固定大小的窗口逐步滑动并进行处理的技术。该算法特别适用于需要在数据流中执行动态窗口操作的场景，如最大子数组、子串查找、滑动窗口平均值等问题。

在应用滑动窗口算法进行久坐行为检测时，首先应去除开头和结尾的睡眠数据，以确保算法聚焦于有效的行为数据区间，从而提高检测准确性。

5.3 久坐预警

基于滑动窗口算法对附件 2 中数据进行久坐预警，结果如表 10 所示：

表 10 久坐预警表

Table 10 Sedentary Warning Table

Pid	Time	Alert
P101	2016-06-23 05:11:19.990000	1
	2016-06-24 00:57:59.990000	1
P102	-	0
P103	2016-12-19 20:39:19.990000	1
	2016-12-20 01:14:19.990000	1
P104	-	0
P105	2016-11-02 03:47:19.990000	1
	2016-11-02 04:18:59.990000	1

Pid	Time	Alert
	2016-11-02 05:03:59.990000	1
	2016-11-02 05:33:59.990000	1
	2016-11-02 07:37:19.990000	1
	2016-11-02 08:57:19.990000	1
	2016-11-02 09:38:59.990000	1
	2016-11-02 10:37:19.990000	1
	2016-11-02 11:07:19.990000	1
P106	2016-05-01 05:26:59.990000	1
	2016-05-01 14:15:19.990000	1
	2016-05-01 15:28:39.990000	1
	2016-05-01 16:48:39.990000	1
	2016-05-01 22:21:59.990000	1
P107	2016-03-20 04:07:39.990000	1
	2016-03-20 04:37:39.990000	1
P108	2016-07-21 04:29:39.990000	1
	2016-07-21 19:56:19.990000	1
P109	2016-07-31 13:13:59.990000	1
	2016-07-31 17:02:19.990000	1
	2016-07-31 17:35:39.990000	1
	2016-07-31 18:57:19.990000	1
	2016-07-31 22:20:39.990000	1
P110	2016-11-08 10:50:59.990000	1
	2016-11-08 11:29:19.990000	1
	2016-11-08 12:44:19.990000	1
	2016-11-08 13:14:19.990000	1
	2016-11-08 23:17:39.990000	1
	2016-11-09 01:32:39.990000	1

Pid	Time	Alert
P111	2016-11-13 14:17:59.990000	1
	2016-11-13 18:22:59.990000	1
	2016-11-13 22:09:39.990000	1
	2016-11-13 22:39:39.990000	1
	2016-11-13 23:09:39.990000	1
	2016-11-13 23:59:39.990000	1
	2016-11-14 00:29:39.990000	1
	2016-11-14 01:56:19.990000	1
	2016-11-14 02:27:59.990000	1
P112	2016-06-04 05:27:39.990000	1
P113	2016-10-12 05:13:59.990000	1
	2016-10-12 09:57:19.990000	1
	2016-10-12 10:27:19.990000	1
	2016-10-12 12:03:59.990000	1
	2016-10-12 12:42:19.990000	1
	2016-10-12 21:08:59.990000	1
	2016-10-12 21:38:59.990000	1
	2016-10-12 22:08:59.990000	1
	2016-10-12 22:47:19.990000	1
	2016-10-12 23:17:19.990000	1
	2016-10-12 23:47:19.990000	1
	2016-10-13 00:17:19.990000	1
	2016-10-13 01:37:19.990000	1
P114	2016-07-26 07:51:19.990000	1
	2016-07-26 09:52:59.990000	1
	2016-07-26 11:17:59.990000	1
	2016-07-26 13:07:59.990000	1

Pid	Time	Alert
	2016-07-26 17:37:59.990000	1
P115	2016-04-23 04:42:39.990000	1
	2016-04-23 17:20:59.990000	1
P116	2016-06-10 04:58:59.990000	1
	2016-06-10 09:33:59.990000	1
	2016-06-10 14:12:19.990000	1
	2016-06-10 15:32:19.990000	1
	2016-06-10 16:23:59.990000	1
	2016-06-10 16:53:59.990000	1
P117	2016-05-13 05:06:59.990000	1
	2016-05-13 05:36:59.990000	1
	2016-05-13 06:06:59.990000	1
	2016-05-13 06:36:59.990000	1
	2016-05-13 07:06:59.990000	1
	2016-05-13 08:20:19.990000	1
	2016-05-13 08:50:19.990000	1
	2016-05-14 00:20:19.990000	1
P118	2016-01-27 05:39:39.990000	1
P119	2016-05-10 14:13:19.990000	1
	2016-05-10 15:04:59.990000	1
	2016-05-10 15:34:59.990000	1
	2016-05-10 16:04:59.990000	1
	2016-05-10 21:48:19.990000	1
	2016-05-10 23:53:19.990000	1
P120	2016-02-26 20:35:59.990000	1
	2016-02-26 21:59:19.990000	1
	2016-02-26 23:52:39.990000	1

六、结语

6.1 模型的优点与缺点

（1）模型的优点：

在问题一中，通过将多进程与 Polars 结合，显著提高了数据处理的运行效率。与传统的 Pandas 相比，理论上实现了 38 倍的速度提升，这使得在处理大规模数据时，不仅大幅缩短了计算时间，也提升了整个分析流程的效率和响应速度。

在问题二中，基于对加速度区间规律的深入分析，构建了每一段区间的统计特征和频域特征，并结合 XGBoost 模型进行预测，极大地增强了模型对加速度数据中潜在模式的识别能力。同时，采用睡眠匹配技术对睡眠数据进行了有效补充，从而减小了模型预测的影响。

在问题三中，通过差分方法和自适应阈值算法，成功识别了睡眠状态的转变。差分方法有效提取了数据中的变化趋势，而自适应阈值的引入进一步优化了睡眠状态识别的准确性，使得模型能够在动态变化的睡眠数据中精确捕捉到状态转换，从而提高了模型的精确度和适应性。

在问题四中，采用滑动窗口算法对久坐行为进行实时预警。该算法能够持续监控用户的行为变化，当久坐阈值被触及时及时发出预警信号。滑动窗口算法凭借其较强的时序性和灵活性，能够捕捉用户在特定时间段内的活动模式，显著提高了预警系统的响应速度和准确性，避免了传统方法可能存在的延迟反应问题。

（2）模型的缺点：

在问题一中，理论测试中 Dask 的应用存在一定不足，未能充分展示其在分布式计算中的优势。虽然 Dask 有潜力提升计算性能，但在当前的测试中未能充分发挥其优势，导致实际效果未达到预期。

在问题二中，特征选择存在一定的合理性问题。部分特征的选择可能未能完全捕捉到加速度数据的关键规律，从而影响了预测结果的准确性，尤其是在高 MET 值预测中的表现仍有提升空间，模型的准确度亟需改进。

在问题三中，睡眠匹配算法的合理性需要进一步验证。尽管该算法在一定程度上有效补充了缺失的睡眠数据，但其假设和匹配策略可能存在一定局限性。为确保更为准确的睡眠状态识别，仍需对其适应性和精度进行深入探讨与优化。

在问题四中，连续两个预警的处理方式尚有优化空间。当前的处理策略未能有效合并相邻的预警信号，导致预警系统在实际应用中存在一定的冗余。

6.2 模型的改进与未来展望

由于当前主流的 Python 数据分析库大多选择与 Pandas 进行兼容或集成，在某些场景下使用 Polars 时需要将数据转换为 Pandas 格式，这一转换过程可能会导致性能下降，从而降低运行速度。这种格式转换不仅增加了额外的计算开销，

还可能影响数据处理的效率，尤其是在处理大规模数据时，可能无法充分发挥 Polars 的高效性能。

在具体应用场景中，如在进行 MET 值估计时，可以通过综合多种特征进行更为精准的预测。此外，在睡眠阶段识别过程中，基于准确的开始记录时间，可以精确地划分不同的睡眠阶段，从而实现更高精度的睡眠分析。在久坐预警系统中，合理地设定预警机制，确保系统在非睡眠时间段内进行有效的久坐监测，并发出及时预警，能够更好地帮助用户保持健康的生活方式。

七、参考文献

[1] Narayanan P K. Data Wrangling using Rust's Polars[M]//Data Engineering for Machine Learning Pipelines: From Python Libraries to ML Pipelines and Cloud Platforms. Berkeley, CA: Apress, 2024: 93-131.

八、附录

附录
介绍：程序文件列表
程序 ├──附件 1 │　　Duration.csv │　　Features.csv │　　lightgbm_model.pkl │　　Merge.csv │　　Metadata1.csv │　　p1-anomaly.py │　　p1-contrast.py │　　p1-duplicate.py │　　p1-file-size.py │　　p1-main-test.py │　　p1-main.py │　　p1-miss.py │　　p1-multi-polars.py │　　p1-polars-test.py │　　p1-polars-visual1.py │　　p1-polars-visual2.py │　　p1-polars-visual3.py

```
│       p1-principle.py
│       p1-regularity.py
│       p1-simple.py
│       p1-statistics.py
│       p1-time-diff.py
│       p2-anomaly.py
│       p2-count.py
│       p2-duration.py
│       p2-features.py
│       p2-index.py
│       p2-merge.py
│       p2-meta-visual.py
│       p2-meta.py
│       p2-pre-Light.py
│       p2-pre-RF.py
│       p2-pre-visual.py
│       p2-pre-XGB.py
│       p2-predict.py
│       p3-main.py
│       p3-show.py
│       p3-sleep-duration.txt
│       p3-visual.py
│       result_1.xlsx
│       xgboost_model.pkl
│
└──附件 2
        Duration.csv
        Metadata1.csv
        Metadata1.json
        Metadata2.csv
        Metadata2.json
        p2-fill.py
        p2-main.py
        p2-merge.py
        p2-pairing.py
        p2-pairing.txt
        p2-predict.py
        p2-retrieve.py
        p2-start.py
        p4-main.py
        p4-result.py
        Result.txt
```