# 基于MetaBCI的人工智能睡眠分期工具

代码使用手册

迈动数康科技有限公司

### **目录**

[基于MetaBCI的人工智能睡眠分期工具 1](#_Toc30993)

[目录 1](#_Toc5353)

[1. 引言 2](#_Toc25064)

[2. 数据集导入及其预处理 3](#_Toc15303)

[2.1. 数据集sleep\_telemetry（brainda.datasets.sleep\_telemetry） 3](#_Toc17472)

[2.1.1. Class Sleep\_telemetry\_data 3](#_Toc7815)

[2.1.2. Class Sleep\_telemetry 3](#_Toc21196)

[2.2. 数据集sleep\_cassette（brainda.datasets.sleep-cassette） 4](#_Toc21023)

[2.3. 数据集sleep\_shhs（brainda.datasets.sleep-shhs） 5](#_Toc3957)

[2.4. 数据集sleep\_apple(brainda.datasets.sleep-apple) 6](#_Toc13941)

[2.5. 基础类class base（brainda.datasets.base） 6](#_Toc12704)

[3. 深度学习模型(brainda.algorithms.deep\_learning） 7](#_Toc2060)

[3.1. 睡眠分期模型Attnsleep(/AttnSleep.py) 7](#_Toc9135)

[3.1.1. class AttnSleep 7](#_Toc20483)

[3.1.2. class AttnSleep\_1CH 7](#_Toc31104)

[3.1.3. class AttnSleep\_2CH 7](#_Toc11536)

[3.1.4. class AttnSleep\_3CH 8](#_Toc4088)

[3.1.5. class SkorchN\_sleep 8](#_Toc1381)

[3.1.6. class BestEpochCallback 8](#_Toc17889)

[3.2. 基础类base(/base.py) 8](#_Toc28825)

[3.2.1. class NeuralNetClassifierNoLog 8](#_Toc153)

[3.2.2. class AvgPool1dWithConv 8](#_Toc25107)

[4. 运行演示（demo\_sleep） 9](#_Toc874)

[4.1. 模型训练 9](#_Toc8067)

[4.1.1. eegnet（/eegnet-run.py） 9](#_Toc5770)

[4.1.2. Attnsleep (/Attnsleep\_run.py) 10](#_Toc11375)

[4.2. 交叉验证（/cross\_validation.py） 11](#_Toc8966)

[4.3. 预测（/predict.py） 11](#_Toc23263)

[4.3.1. 预测结果（/predict.py） 11](#_Toc21292)

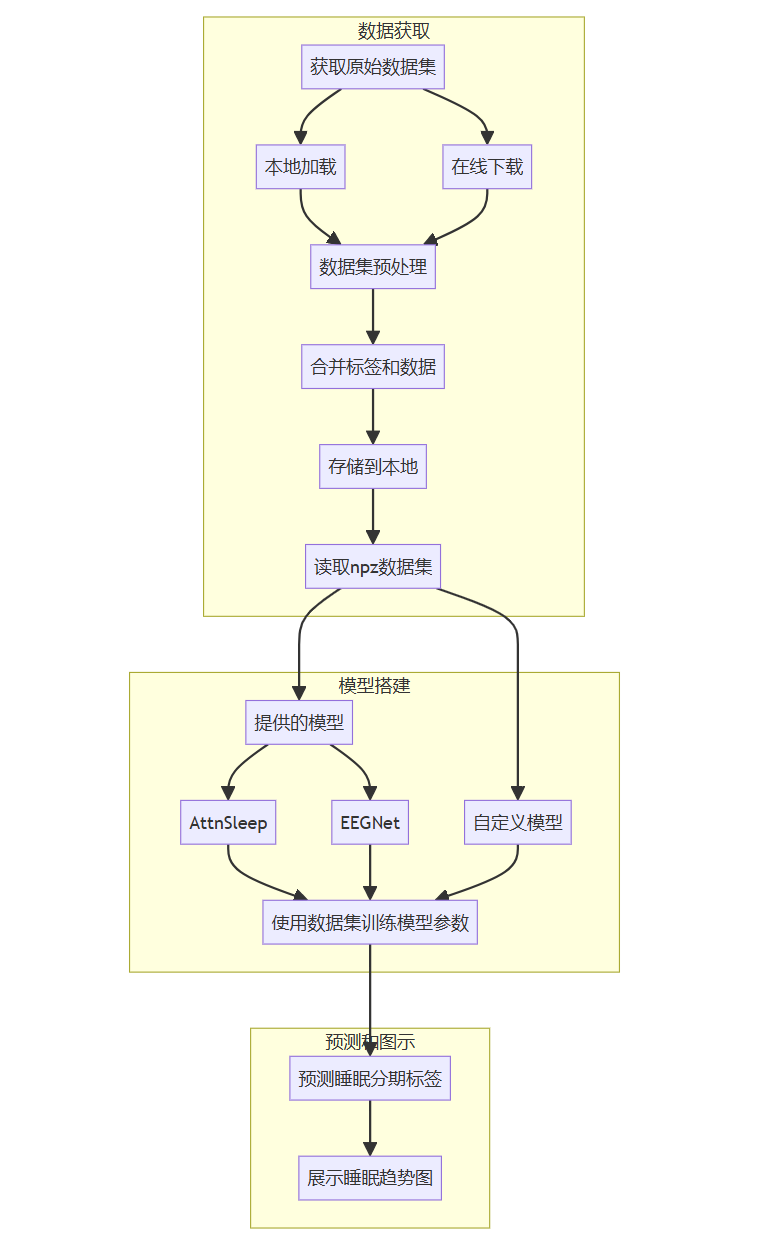
[4.3.2. 图示（/show.py） 11](#_Toc29352)

[4.3.3. 预测结果平滑处理（/smooth.py） 11](#_Toc823)

### 引言

MetaBCI是一款针对BCI的开源平台。本项目的宗旨是在MetaBCI的框架上添加睡眠分期数据集的获取与处理，睡眠分期深度学习模型，训练方法和预测方法，最后实现结果可视化，对部分功能的原基类进行调整以更好的适应睡眠分期任务。

基于MetaBCI的人工智能睡眠分期的流程图如下：



在参考代码使用案例时，请注意替换路径yourpath以及自行导入模块，并确保模型通道数和数据通道数的匹配性。

### 数据集导入及其预处理

##### 数据集sleep\_telemetry（brainda.datasets.sleep\_telemetry）

###### Class Sleep\_telemetry\_data

该类可以通过在线下载或者本地加载的方式获取睡眠分期数据sleep-telemetry的数据，但**无需实例化这个类**，子类Sleep\_telemetry已经包含了这些功能，只需要实例化子类即可。

Methods

***data\_path\_url*** (subject, path=None, force\_update=False, update\_path=None, proxies=None, verbose=None):

**在线下载**原始数据，并返回存储的路径。

***data\_path*** (subject, path=None, force\_update=False, update\_path=None, proxies=None, verbose=None):

**本地已经存储**数据，输入文件夹的路径和需要的对象，返回路径。

***\_get\_single\_subject\_data*** (subject, verbose = None)

内部方法，返回单个数据

***get\_freq*** (subject)

获取采样率

***read\_data\_name*** （line\_number）

获取数据文件名和标签文件名

***count\_npz\_files*** （count\_npz\_files）

查找所有npz文件，返回文件个数

###### Class Sleep\_telemetry

该类继承了Sleep\_telemetry\_data并且可以获取睡眠分期数据sleep-telemetry的标签，将数据和标签合并为数据集，对数据集的整合和转码，将处理后的数据集存储和读取。

Parameters

subjects=list(range(43)) # 共44个试次

channels=['Fpz-Cz','Pz-Oz']

# 在线下载的地址

sleep\_URL = "https://physionet.org/files/sleep-edfx/1.0.0/sleep-telemetry/"

events={"Sleep stage W": 0,"Sleep stage 1": 1,"Sleep stage 2": 2,"Sleep stage 3": 3,"Sleep stage 4": 3,"Sleep stage R": 4,"Sleep stage ?": 5,"Movement time": 5}

Methods

***label\_path\_url*** (subject, path=None, force\_update=False, update\_path=None, proxies=None, verbose=None):

**在线下载**原始标签，并返回存储的路径（需要和数据保存在一个路径下）。

***label\_path*** (subject, path=None, force\_update=False, update\_path=None, proxies=None, verbose=None):

本地已经存储标签，输入文件夹的路径和需要的对象，返回路径。

***\_get\_single\_subject\_label*** (subject, verbose = None)

内部方法，返回单个对象的原始标签文件

***get\_label*** (subject, verbose = None)

获取原始标签对象

***save\_processed\_data*** (subjects:List[int], update\_path=None, select\_ch=None)

核心方法，输入通道，对象，对已经存储的原数据集(**如果还未下载将自动下载**)，对原数据集处理后，保存为npz文件在指定路径下。select\_ch默认为["EEG Fpz-Cz"]，另可选择通道EEG Pz-Oz，EOG horizontal。例如，若想获得EEG Fpz-Cz和EEG Pz-Oz数据，

***get\_processed\_data*** (subject, path=None, force\_update=False, update\_path=None, proxies=None, verbose=None，num\_classes = 5

)

update\_path 存储npz文件的地址。

num\_classes 数据分类门数，默认为5，可选2,3,4。

核心方法，读取已经处理的数据文件，需要本地有npz文件（通过save\_processed\_data方法可得），获取数据集[labels,read\_datas]。

Example:

from metabci.brainda.datasets.sleep\_telemetry import Sleep\_telemetry

dataPath = yourpath *# 你的指定路径*

sleep = Sleep\_telemetry(dataPath=dataPath)

sleep.save\_processed\_data(update\_path=dataPath)

data = sleep.get\_processed\_data(update\_path=dataPath)

labels, read\_datas = data[0], data[1]

注：在调用save\_processed\_data时自动实现下载，处理，转存数据。处理好的数据为npz格式。下载的原始数据在指定路径的文件夹下，npz文件在指定路径下。

##### 数据集sleep\_cassette（brainda.datasets.sleep-cassette）

Class SleepCassette

可以获取睡眠分期数据集sleep-cassette。该类继承了类 Sleep\_telemetry，并覆写了部分方法，该类的所有方法与类 Sleep\_telemetry一样。

Parameters

subjects = list(range(152)) # 共44个试次

# 在线下载的地址

sleep\_URL = 'https://physionet.org/files/sleep-edfx/1.0.0/sleep-cassette/'

Example:

from metabci.brainda.datasets.sleep-cassette import SleepCassette

dataPath = yourpath *# 你的指定路径*

sleep =SleepCassette (dataPath=dataPath)

sleep.save\_processed\_data(update\_path=dataPath)

data = sleep.get\_processed\_data(update\_path=dataPath)

labels, read\_datas = data[0], data[1]

##### 数据集sleep\_shhs（brainda.datasets.sleep-shhs）

Class Sleep\_SHHS

可以获取睡眠分期数据集sleep-shhs。但由于SHHS的原数据集并没有公开下载方式，通过以下链接来下载经过筛选的数据集。

<https://pan.baidu.com/s/1_3umu61B8wln_MylCjoDbw?pwd=mdsk>

Parameters

subjects=list(range(49)) # 共50个试次

channels=["EEG", "EOG(L)", "EOG(R)"]

events={"W": 0,"N1": 1,"N2": 2,"N3": 3,"R": 4}

该数据集的基本方法与上面的数据相似。但是没有label\_path\_url和data\_path\_url的在线下载方法。

Example

path = yourpath *# 原始数据存储路径*

dataPath = yourpath *# 处理数据的目标存储路径*

sleep = Sleep\_SHHS(dataPath=path)

sleep.save\_processed\_data(update\_path=dataPath, select\_ch=["EEG", "EOG(L)", "EOG(R)"], subjects=[1])

savepath = yourpath *# 读取npz数据的路径，一般在dataPath 下*

data = sleep.get\_processed\_data(subjects=[0], update\_path=savepath)

labels, read\_datas = data[0], data[1]

print(read\_datas)

##### 数据集sleep\_apple(brainda.datasets.sleep-apple)

Class Sleep\_Apples

可以获取睡眠分期数据集sleep-shhs。但由于SHHS的原数据集并没有公开下载方式，通过以下链接来下载经过筛选的数据集。

<https://pan.baidu.com/s/1_3umu61B8wln_MylCjoDbw?pwd=mdsk>

path = yourpath *# 原始数据存储路径*

dataPath = yourpath *# 处理数据的目标存储路径*

sleep = Sleep\_Apples(dataPath=path)

sleep.save\_processed\_data(update\_path=dataPath, select\_ch=["C3\_M2", "ROC", "LOC"]

, subjects=[1])

savepath = yourpath *# 读取npz数据的路径，一般在dataPath 下*

data = sleep.get\_processed\_data(subjects=[0], update\_path=savepath)

labels, read\_datas = data[0], data[1]

print(read\_datas)

注：所有的数据集的get\_processed\_data方法中的subjects与save\_processed\_data中的subjects并不为同一个subjects。在get\_processed\_data中的subjects编号根据已处理的文件格式决定，例如只处理了3个文件，那么范围就是[0,1,2]。

##### 基础类class base（brainda.datasets.base）

Methods

***save\_processed\_data*** (subjects : list, path, update\_path)

睡眠分期数据集往往繁杂，它们无法通过简单的处理而得到数据和标签。而且这些数据集有很多的试次，这会占用较大的内存，给直接处理数据带来较大的压力。所以在这种情况时我们会选择先下载原始数据，再将原始数据集处理转码成简单的文件保存下来，增加这个方法以供子类去覆写。

***get\_processed\_data*** (subjects : list, update\_path)

配套save\_processed\_data方法，读取文件。

注：我们没有把这些方法写成抽象方法，以便在继承base可以自主选择去覆写这个类

### 深度学习模型(brainda.algorithms.deep\_learning）

##### 睡眠分期模型Attnsleep(/AttnSleep.py)

###### class AttnSleep

已经完成封装的核心类，实现自动睡眠分期的深度学习模型，可兼容1,2,3通道的数据。该模型借鉴了名为 AttnSleep 的基于注意力的新型深度学习架构。我们调整了模型的结构，将MetaBCI中的功能加入到了模型当中，并增加了多通道数据。

[1] E. Eldele et al., "An Attention-Based Deep Learning Approach for Sleep Stage Classification With Single-Channel EEG," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 29, pp. 809-818, 2021, doi: 10.1109/TNSRE.2021.3076234.

data : tensor(epoch, channel, data)

label : tensor(label)

Selects the dual-channel model and sets the number of classes to 5.

Example:

model = AttnSleep(3, 5)

model.fit(data, label)

###### class AttnSleep\_1CH

构成class AttnSleep训练单通道数据的功能。

Params

N = 2 # TCE模型的克隆数量

d\_model = 80 # 用于表示模型中全连接层的神经元个数

d\_ff = 120 # 前馈神经网络的维度，通常用于处理注意力层的输出

h = 5 # 多头注意力机制中的注意力头的数量, 这里代表划分为子空间的个数

dropout = 0.3 # 30%的丢弃率

afr\_reduced\_cnn\_size = 30 # SE block中的通道数

###### class AttnSleep\_2CH

构成class AttnSleep训练双通道数据的功能。参数设置和 AttnSleep\_1CH一样。该模型通过分别处理两个通道的数据，提取特征并编码，再将两个通道的特征合并后进行分类。这种双通道处理方法可以利用两个不同通道的特征信息，提高模型对输入数据的理解能力和分类性能。

###### class AttnSleep\_3CH

构成class AttnSleep训练双通道数据的功能。分别处理三个不同的通道数据，提取各自的特征，然后将这些特征合并，进行进一步的处理和分类。这种多通道的处理方式可以利用不同通道的信息互补，增强模型的整体性能。

###### class SkorchN\_sleep

覆写SkorchNet类，修改类中的参数可以实现调参功能。

optimizer\_weight\_decay=1e-4 L2 正则化参数

batch\_size=128 批次大小

lr=1e-3 学习率

###### class BestEpochCallback

增加功能，在训练结束后可以打印出在测试集上最高准确率的一次epoch。在SkorchNet\_sleep的回调中自动实现。

##### 睡眠分期模型TinySleepNet（/tinysleepnet.py）

##### 基础类base(/base.py)

###### class NeuralNetClassifierNoLog

对原有类进行修改和扩写，使其可以实现传入指定的训练集和测试集的功能，方便后续交叉验证的功能。

Methods

***fit*** （X , y , valid\_data=None）

核心方法，输入数据集实现模型训练的功能，当输入valid\_data时，valid\_data为测试集，格式为[X\_valid, y\_valid]，X , y 为训练集。当不输入valid\_data时，就会自动拆分输入的X , y为训练集的测试集。

**经过测试如果使用 valid\_data的功能可以将训练速度提高一倍左右。**

***get\_split\_datasets*** （X, y=None）

用于支持fit中的valid\_data功能

###### class AvgPool1dWithConv

补充了1d 数据的卷积计算平均池化。

Example

class SELayer(nn.Module):

def \_\_init\_\_(self):

self.avg\_pool = AvgPool1dWithConv(kernel\_size=3, stride=1, dilation=1, padding=1)

### 运行演示（demo\_sleep）

##### 模型训练

###### eegnet（/eegnet-run.py）

使用eegnet模型训练sleep-telemetry的单通道EEG Fpz-Cz数据集。更改subjects来选定数据对象，更改dataPath来选定下载原始数据集的路径，更改path来选定处理后数据集保存的路径。当然，你也可以更改数据集的来源。

训练的结果将保存到通目录下的checkpoints/训练时间/params.pt文件中。

from metabci.brainda.algorithms.deep\_learning import np\_to\_th

from metabci.brainda.algorithms.deep\_learning.eegnet import EEGNet

from metabci.brainda.datasets.sleep\_telemetry import Sleep\_telemetry

dataPath = yourpath

*# Processed npz data address*

path =yourpath

sleep = Sleep\_telemetry(dataPath)

sleep.save\_processed\_data(update\_path=path)

subjects = list(range(1, 6))

data = sleep.get\_processed\_data(update\_path=path, subjects=subjects)

label, input\_data = data[0], data[1]

*# check the shape of data*

model = EEGNet(1, 3000, 5)

X\_train = np\_to\_th(input\_data)

y\_train = np\_to\_th(label)

print(X\_train.shape)

*# The value of the label in the fit function must be in long format.*

y\_train = y\_train.long()

model.fit(X\_train, y\_train)

###### Attnsleep (/Attnsleep\_run.py)

同上，使用Attnsleep模型训练。

from metabci.brainda.algorithms.deep\_learning import np\_to\_th

from metabci.brainda.algorithms.deep\_learning.AttnSleep import AttnSleep

from metabci.brainda.datasets.sleep\_telemetry import Sleep\_telemetry

dataPath = yourpath

path = yourpath

sleep = Sleep\_telemetry(dataPath)

subjects = [0]

data = sleep.get\_processed\_data(update\_path=path, subjects=subjects)

label, input\_data = data[0], data[1]

model = AttnSleep(1, 5)

X\_train = np\_to\_th(input\_data)

y\_train = np\_to\_th(label)

y\_train = y\_train.long()

model.fit(X\_train, y\_train)

Sleep\_telemetry的Fpz-Cz通道数据集训练保存在/checkpoints/ST-Fpz-Cz下

Sleep\_telemetry的Pz-Oz通道数据集训练保存在/checkpoints/ST-Fpz-Oz下

Sleep\_cassette的Fpz-Cz通道数据集训练保存在/checkpoints/SC-Fpz-Cz下

Sleep\_cassette的Pz-Oz通道数据集训练保存在/checkpoints/SC-Fpz-Oz下

##### 交叉验证（/cross\_validation.py）

###### 交叉验证实现

在简单模型训练的基础上，增加了交叉验证的功能，并保存交叉验证的混淆矩阵到confusion\_matrix文件夹中，（使用Attnsleep训练，可以选择通道数和分类数）。

Methods

***cross\_train\_model*** (datas, n\_splits=5,model\_params=(1,5), model\_selection=EnhancedStratifiedKFold)

datas必须是get\_processed\_data获取的[labels,data]的形式。

n\_splits=5交叉验证的折数

model\_params （channel, num\_classes注意输入数据和datas格式匹配。

model\_selection数据折叠方式，从brainda/algorithms/utils/model\_selection选择合适的方法，默认使用EnhancedStratifiedKFold。

Example

npz\_path = yourpath

sleep = SleepCassette()

subjects = list(range(49))

data = sleep.get\_processed\_data(update\_path=npz\_path, subjects=subjects)

cross\_train\_model(data, model\_params=(1, 5))

###### 混淆矩阵图示（/show\_confusion.py）

Methods

***plot\_confusion\_matrix*** (readPath:str, labels:List)

readPath (str): 混淆矩阵文件的路径

labels (list of str): 类别标签列表

Example

readPath = yourpath

labels = ['Wake', 'N1', 'N2', 'N3', 'REM']

plot\_confusion\_matrix(readPath, labels)

注：必须要完整运行交叉验证后得到混淆矩阵文件才能使用此方法。

##### 预测（/predict.py）

###### 预测结果（/predict.py）

Methods

***save\_res\_pre*** （X: np.ndarray, parampath: str, train\_selection=AttnSleep) -> np.ndarray

输入数据，返回预测标签，并保存文件

###### 图示（/show.py）

Methods

***plotAnalyze*** (data : np.ndarray)

生成一张饼图，显示睡眠分期各阶段的比例。

***plotTime*** (ax, data: np.ndarray, flag\_modi= False, color="blue", name: str ="")

根据输入的标签绘制睡眠分期趋势图像。

###### 预测结果平滑处理（/smooth.py）

Methods

***Smooth*** (sleep\_stages: np.ndarray) -> np.ndarray:

对模型预测的标签做平滑性处理，其中的平滑处理规则参考了下面的文献，经过实际测试可以使睡眠分期的准确率平均提高2%左右。

1. S. -F. Liang, C. -E. Kuo, Y. -H. Hu and Y. -S. Cheng, "A rule-based automatic sleep staging method," 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Boston, MA, USA, 2011, pp. 6067-6070, doi: 10.1109/IEMBS.2011.6091499.
2. Berry RB, Brooks R, Gamaldo C, Harding SM, Lloyd RM, Quan SF, Troester MT, Vaughn BV. AASM scoring manual updates for 2017 (Version 2.4). J Clin Sleep Med. 2017;13(5):665–666.

最后把预测，平滑处理，图示，评估结合起来综合演示。

Example

performance = Performance(estimators\_list=["Acc", "TPR", "FNR", "TNR"])

datapath = yourpath

parampath =yourpath

sleep\_data = Sleep\_telemetry()

datas = sleep\_data.get\_processed\_data(subjects=[14], update\_path=datapath)

y\_predict = save\_res\_pre(datas[1], parampath)

y\_true = datas[0]

y\_predict\_sm = smooth(y\_predict)

print(performance.evaluate(y\_true, y\_predict))

print(performance.evaluate(y\_true, y\_predict\_sm))

plotAnalyze(y\_predict\_sm)

fig, axs = plt.subplots(3, 1, figsize=(15, 10))

plotTime(axs[0], y\_true, flag\_modi=False, color="black", name="PSG true label")

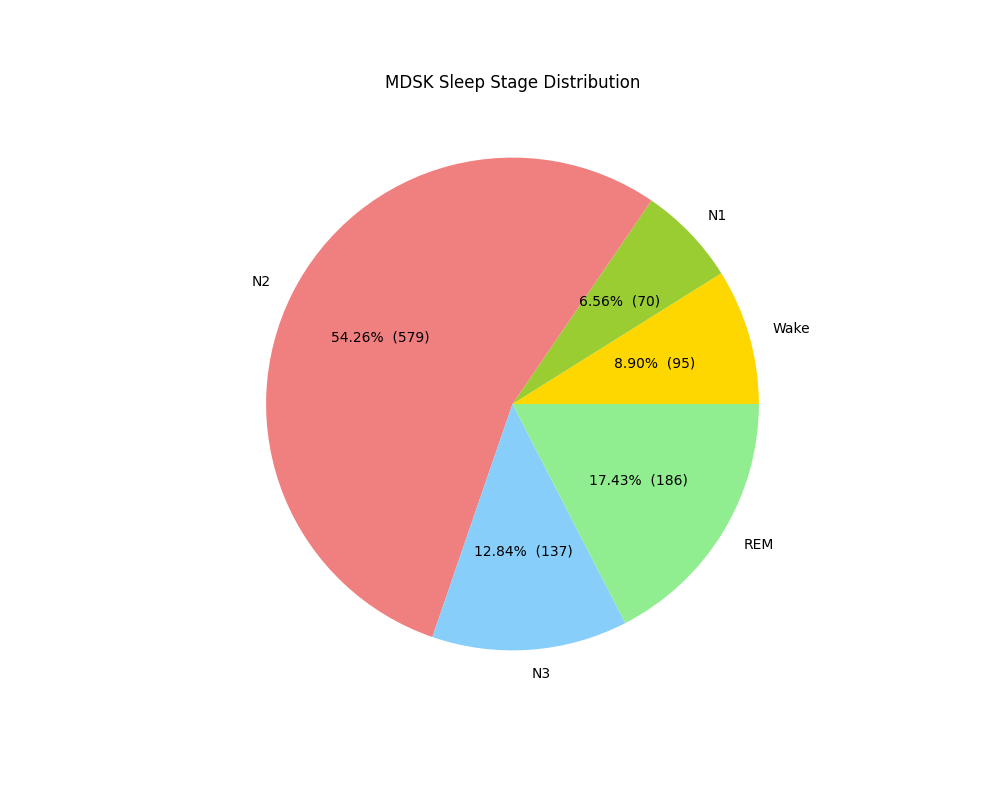
plotTime(axs[1], y\_predict, flag\_modi=False, color="GoldenRod", name="prediction")

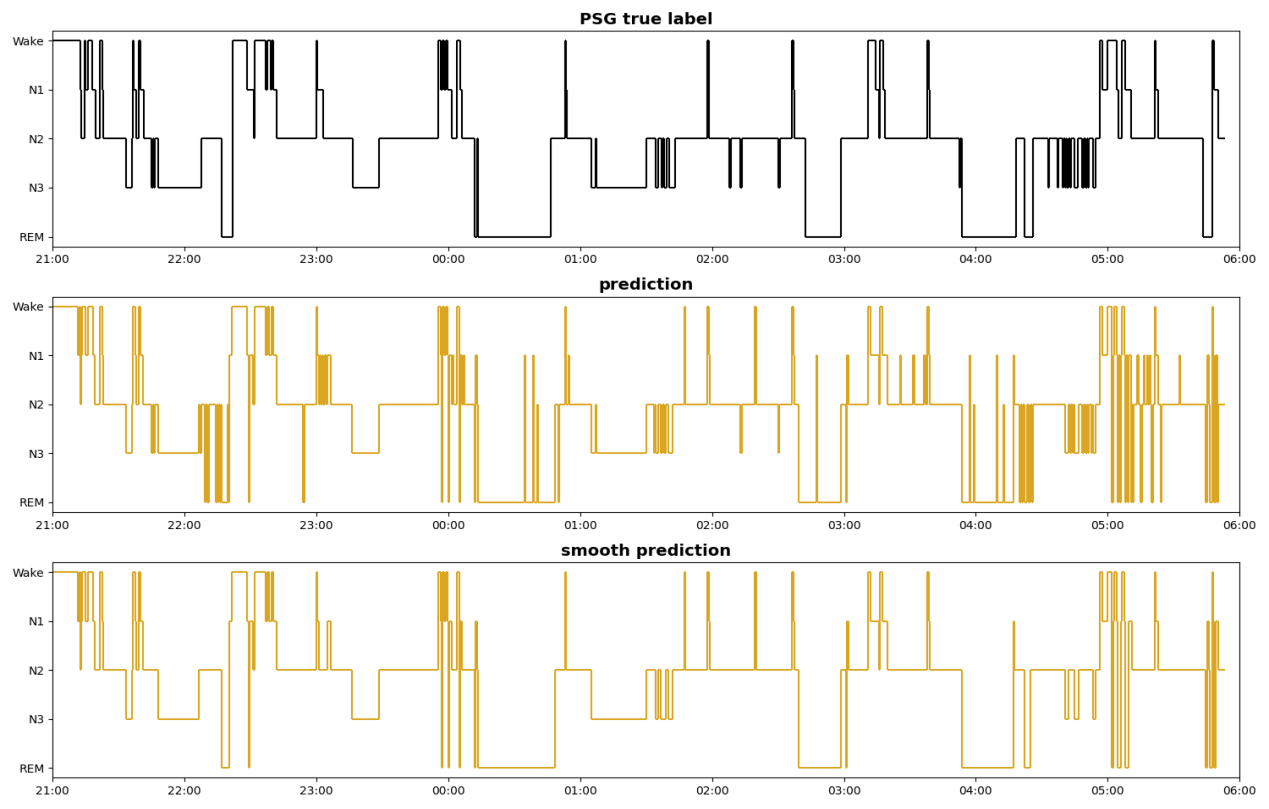
plotTime(axs[2], y\_predict\_sm, flag\_modi=False, color="GoldenRod", name="smooth prediction")

plt.tight\_layout()

plt.show()

图示结果如下所示：





###### 