## 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。 |

### 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  # 零样本提示分类代码  def zero\_shot\_classification():  # 读取数据  data = pd.read\_csv('Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv')  features = ['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration', 'Quality of Sleep',  'Physical Activity Level', 'Stress Level', 'BMI Category', 'Blood Pressure', 'Heart Rate',  'Daily Steps']  X = data[features]  # 简单的预定义规则分类（这里只是示例，实际需要更复杂的规则）  def simple\_rule\_based\_classification(row):  if row['Sleep Duration'] < 6 and row['Quality of Sleep'] < 5:  return 'Insomnia'  elif row['Heart Rate'] > 80 and row['Sleep Duration'] > 8:  return 'Sleep Apnea'  else:  return 'normal'  data['Sleep Disorder'] = X.apply(simple\_rule\_based\_classification, axis=1)  data.to\_csv('classfiers\_by\_90samples.csv', index=False)  # 90样本提示分类代码  def ninety\_shots\_classification():  train\_data = pd.read\_csv('Sleep\_health\_and\_lifestyle\_dataset\_selected\_90.csv')  test\_data = pd.read\_csv('Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv')  features = ['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration', 'Quality of Sleep',  'Physical Activity Level', 'Stress Level', 'BMI Category', 'Blood Pressure', 'Heart Rate',  'Daily Steps']  X\_train = train\_data[features]  y\_train = train\_data['Sleep Disorder']  X\_test = test\_data[features]  from sklearn.linear\_model import LogisticRegression  model = LogisticRegression()  model.fit(X\_train, y\_train)  y\_pred = model.predict(X\_test)  test\_data['Sleep Disorder'] = y\_pred  test\_data.to\_csv('classfiers\_by\_90samples.csv', index=False)  # 分解提示分类代码  def decomposed\_classification():  train\_data = pd.read\_csv('Sleep\_health\_and\_lifestyle\_dataset\_selected\_90.csv')  test\_data = pd.read\_csv('Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90\_without\_last\_column.csv')  features = ['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration', 'Quality of Sleep',  'Physical Activity Level', 'Stress Level', 'BMI Category', 'Blood Pressure', 'Heart Rate',  'Daily Steps']  X\_train = train\_data[features]  y\_train = train\_data['Sleep Disorder']  X\_test = test\_data[features]  from sklearn.model\_selection import GridSearchCV  from sklearn.svm import SVC  from sklearn.linear\_model import LogisticRegression  from sklearn.tree import DecisionTreeClassifier  from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier  from sklearn.neighbors import KNeighborsClassifier  models = {  'Logistic Regression': LogisticRegression(),  'Decision Tree': DecisionTreeClassifier(),  'Random Forest': RandomForestClassifier(),  'Gradient Boosting': GradientBoostingClassifier(),  'SVM': SVC(),  'K-Nearest Neighbors': KNeighborsClassifier()  }  best\_model = None  best\_score = 0  for model\_name, model in models.items():  if model\_name == 'SVM':  param\_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf', 'poly']}  elif model\_name == 'Random Forest':  param\_grid = {'n\_estimators': [50, 100, 150], 'max\_depth': [None, 5, 10]}  elif model\_name == 'Gradient Boosting':  param\_grid = {'n\_estimators': [50, 100, 150], 'learning\_rate': [0.01, 0.1, 0.2]}  elif model\_name == 'K-Nearest Neighbors':  param\_grid = {'n\_neighbors': [3, 5, 7]}  else:  param\_grid = {}  if param\_grid:  grid\_search = GridSearchCV(model, param\_grid, cv=5, scoring='accuracy')  grid\_search.fit(X\_train, y\_train)  model = grid\_search.best\_estimator\_  model.fit(X\_train, y\_train)  score = model.score(X\_train, y\_train)  if score > best\_score:  best\_score = score  best\_model = model  y\_pred = best\_model.predict(X\_test)  test\_data['Sleep Disorder'] = y\_pred  test\_data.to\_csv('classfiers\_by\_90samples.csv', index=False)  # 评估代码  def evaluate\_classification():  true\_data = pd.read\_csv('Sleep\_health\_and\_lifestyle\_dataset\_remaining\_90.csv')  pred\_data = pd.read\_csv('classfiers\_by\_90samples.csv')  y\_true = true\_data['Sleep Disorder']  y\_pred = pred\_data['Sleep Disorder']  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')  print(f'Accuracy: {accuracy}')  print(f'Precision: {precision}')  print(f'Recall: {recall}')  print(f'F1 Score: {f1}')  cm = confusion\_matrix(y\_true, y\_pred)  print('Confusion Matrix:')  print(cm)  from sklearn.preprocessing import LabelEncoder  le = LabelEncoder()  y\_true\_encoded = le.fit\_transform(y\_true)  y\_pred\_encoded = le.transform(y\_pred)  fpr = {}  tpr = {}  roc\_auc = {}  n\_classes = len(le.classes\_)  for i in range(n\_classes):  fpr[i], tpr[i], \_ = roc\_curve(y\_true\_encoded == i, y\_pred\_encoded == i)  roc\_auc[i] = auc(fpr[i], tpr[i])  plt.figure()  for i in range(n\_classes):  plt.plot(fpr[i], tpr[i], label=f'Class {le.classes\_[i]} (AUC = {roc\_auc[i]:.2f})')  plt.plot([0, 1], [0, 1], 'k--')  plt.xlim([0.0, 1.0])  plt.ylim([0.0, 1.05])  plt.xlabel('False Positive Rate')  plt.ylabel('True Positive Rate')  plt.title('Receiver Operating Characteristic')  plt.legend(loc="lower right")  plt.show()  # 运行示例  # zero\_shot\_classification()  # ninety\_shots\_classification()  # decomposed\_classification()  # evaluate\_classification() |

上述代码实现了基于不同提示策略的睡眠障碍分类及评估功能。zero\_shot\_classification函数通过简单的预定义规则进行零样本提示分类；ninety\_shots\_classification函数使用 90 样本提示策略，基于逻辑回归模型进行分类；decomposed\_classification函数采用分解提示策略，尝试多种分类器并进行参数调优，最终选择性能最佳的分类器进行分类；evaluate\_classification函数用于评估分类结果，计算准确率、精确率、召回率、F1 分数、混淆矩阵并绘制 ROC 曲线。