def all\_features(X):  
  
 R1 = X['c2h2'] / (X['c2h2'] + X['ch4'] + X['c2h6'])  
 # R2= X['ch4'] / (X['c2h2'] + X['ch4'] + X['c2h6'])  
 # R3 = X['c2h6'] / (X['c2h2'] + X['ch4'] + X['c2h6'])  
 R4 = X['c2h4'] / (X['c2h6'])  
 R5 = X['c2h2'] / (X['c2h4'])  
 R6 = X['c2h2'] / (X['h2'] + X['ch4'] + X['c2h2'] + X['c2h4'] + X['c2h6'])  
 R7 = X['c2h2'] / (X['ch4'])  
 R8 = (X['c2h2']+X['ch4']) / (X['h2'] )  
 # R9 = X['h2'] / (X['c2h2'])  
 # R10 = X['c2h2'] / (X['c2h4'] + X['ch4'] + X['c2h6'])  
 # R11 = (X['ch4'] + X['c2h6']) / (X['h2'] + X['ch4'] + X['c2h2'] + X['c2h4'] + X['c2h6'])  
 # E\_H = X['energy\_index'] = (X['h2'] \* 1 + X['ch4'] \* 2 + X['c2h6'] \* 3 + X['c2h4'] \* 4 + X['c2h2'] \* 5)  
 # # E\_L = ( X['h2'] \* 1 + X['ch4'] \* 2 + X['c2h6'] \* 3)  
 E = (X['h2'] \* 1 + X['ch4'] \* 2 + X['c2h6'] \* 3 + X['c2h4'] \* 4 + X['c2h2'] \* 5)  
 # 处理异常值  
 R1 = R1.replace([np.inf, -np.inf], 0).fillna(0)  
 R4 = R4.replace([np.inf, -np.inf], 0).fillna(0)  
 R5 = R5.replace([np.inf, -np.inf], 0).fillna(0)  
 R6 = R6.replace([np.inf, -np.inf], 0).fillna(0)  
 R7 = R7.replace([np.inf, -np.inf], 0).fillna(0)  
 R8 = R8.replace([np.inf, -np.inf], 0).fillna(0)  
   
 E = E.replace([np.inf, -np.inf], 0).fillna(0)  
 # 添加到特征矩阵  
 X['R1: C2H2/(C2H2+CH4+C2H6)'] = R1  
 X['R4: C2H4/C2H6'] = R4  
 X['R5: C2H2/C2H4'] = R5  
 X['R6: C2H2/(H2+CH4+C2H2+C2H4+C2H6)'] = R6  
 X['R7: C2H2/CH4'] = R7  
 X['R8: (C2H2+CH4)/H2'] = R8  
  
 return X  
  
all\_train\_data = all\_features(train\_all)  
all\_test\_data = all\_features(test\_all)

X\_train = all\_train\_data.drop('act', axis=1)  
y\_train = all\_train\_data['act']  
X\_test = all\_test\_data.drop('act', axis=1)  
y\_test = all\_test\_data['act']  
  
# 标签映射（可选）  
label\_mapping = {  
 1: 'HED',  
 2: 'HT',  
 3: 'LED',  
 4: 'LT',  
 5: 'MT',  
 6: 'PD'  
}  
target\_names = [label\_mapping[i] for i in sorted(label\_mapping.keys())]  
  
# 修改 train\_and\_evaluate 函数，返回分类报告和 overall metrics  
def train\_and\_evaluate(X\_train, y\_train, X\_test, y\_test):  
 model = RandomForestClassifier(n\_estimators=100, criterion='gini', random\_state=42)  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
 report\_dict = classification\_report(y\_test, y\_pred, target\_names=target\_names, output\_dict=True, zero\_division=0)  
  
 report\_df = pd.DataFrame(report\_dict).transpose()  
  
 return report\_df  
  
results = OrderedDict()  
results['No Oversampling'] = train\_and\_evaluate(X\_train, y\_train, X\_test, y\_test)  
  
resamplers = {  
 'SMOTE': SMOTE(random\_state=42),  
}  
  
for name, sampler in resamplers.items():  
 X\_resampled, y\_resampled = sampler.fit\_resample(X\_train, y\_train)  
 results[name] = train\_and\_evaluate(X\_resampled, y\_resampled, X\_test, y\_test)  
  
print("\n=== 每类指标报告 ===")  
for method, report\_df in results.items():  
 print(f"\n[{method}]")  
 print(report\_df.loc[target\_names][['precision', 'recall', 'f1-score']].round(4))

# 应用 SMOTE  
smote = SMOTE(random\_state=42)  
X\_resampled, y\_resampled = smote.fit\_resample(X\_train, y\_train)  
  
# 训练模型  
model = RandomForestClassifier(n\_estimators=100, criterion='gini', random\_state=42)  
model.fit(X\_resampled, y\_resampled)  
  
# 模型预测并评估  
y\_pred = model.predict(X\_test)  
explainer = shap.TreeExplainer(model)  
shap\_values = explainer.shap\_values(X\_test)  
  
for i, class\_name in enumerate(model.classes\_):  
 #plt.title(f"SHAP Summary Plot for Class {class\_name}", fontsize=15)  
 shap.summary\_plot(shap\_values[:, :, i], X\_test, class\_names=[class\_name], title=f"SHAP Summary Plot for Class {class\_name}")  
   
 plt.show()