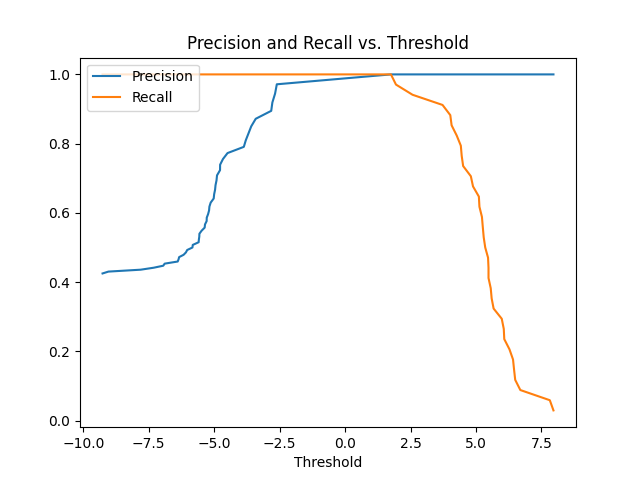
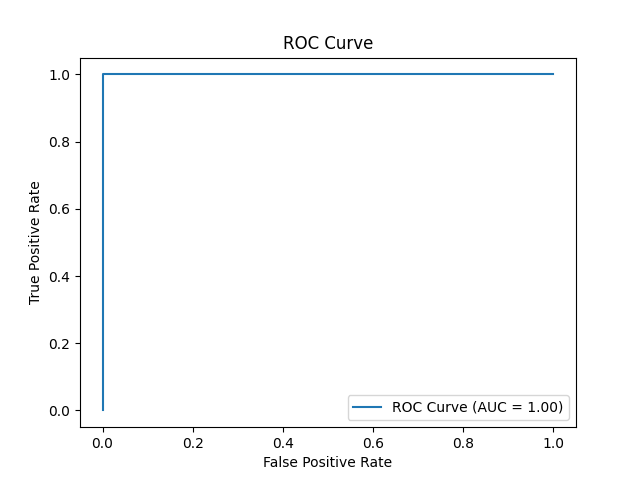
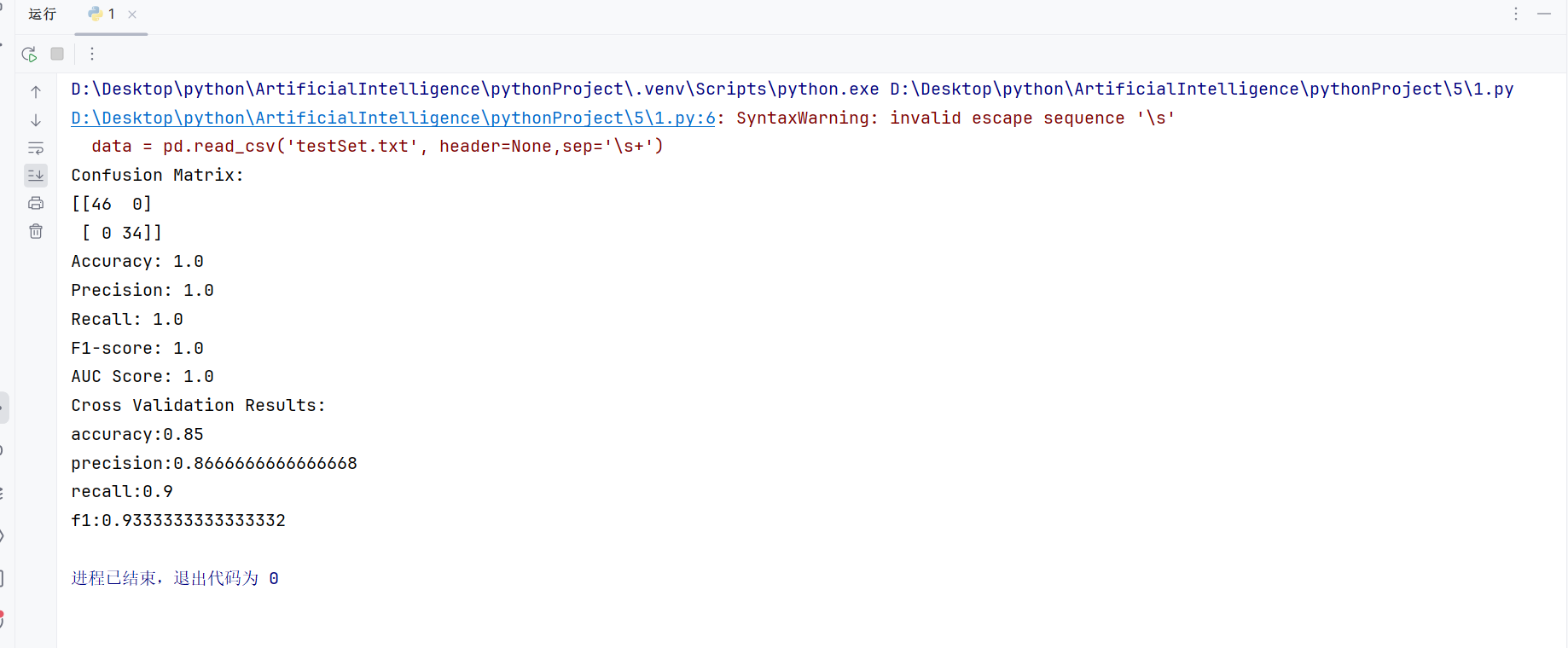
1、读取testSet.txt数据，划分训练集和测试集。

（1）使用逻辑回归模型LogisticRegression训练数据，输出训练集的混淆矩阵、准确率、精确率、召回率、F1-score，画精确率和召回率相对于阈值的函数图，绘制ROC曲线图，输出AUC面积。

（2）使用随机梯度下降分类模型SGDClassifier训练数据，基于SGDClassifier模型使用五折交叉验证输出测试集的准确率、精确率、召回率、F1-score。

import numpy as np  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
  
# 读取数据  
data = pd.read\_csv('testSet.txt', header=None,sep='\s+')  
X = data.iloc[:, :-1] # 特征  
y = data.iloc[:, -1] # 标签  
# print(X)  
# print(y)  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_curve, auc, \  
 precision\_recall\_curve  
import matplotlib.pyplot as plt  
  
# 训练逻辑回归模型  
log\_reg\_model = LogisticRegression()  
log\_reg\_model.fit(X\_train, y\_train)  
  
# 在训练集上做预测  
y\_train\_pred = log\_reg\_model.predict(X\_train)  
  
# 计算混淆矩阵  
conf\_matrix = confusion\_matrix(y\_train, y\_train\_pred)  
print("Confusion Matrix:")  
print(conf\_matrix)  
  
# 计算准确率  
accuracy = accuracy\_score(y\_train, y\_train\_pred)  
print("Accuracy:", accuracy)  
  
# 计算精确率  
precision = precision\_score(y\_train, y\_train\_pred)  
print("Precision:", precision)  
  
# 计算召回率  
recall = recall\_score(y\_train, y\_train\_pred)  
print("Recall:", recall)  
  
# 计算F1-score  
f1 = f1\_score(y\_train, y\_train\_pred)  
print("F1-score:", f1)  
  
# 绘制精确率和召回率相对于阈值的函数图  
y\_scores = log\_reg\_model.decision\_function(X\_train)  
precisions, recalls, thresholds = precision\_recall\_curve(y\_train, y\_scores)  
plt.plot(thresholds, precisions[:-1], label="Precision")  
plt.plot(thresholds, recalls[:-1], label="Recall")  
plt.xlabel("Threshold")  
plt.legend(loc="upper left")  
plt.title("Precision and Recall vs. Threshold")  
plt.show()  
  
# 绘制ROC曲线图  
# 计算 ROC 曲线的参数  
fpr, tpr, thresholds = roc\_curve(y\_train, y\_scores)  
# 计算 AUC  
roc\_auc = auc(fpr, tpr)  
  
# 画 ROC 曲线图  
plt.plot(fpr, tpr, label="ROC Curve (AUC = {:.2f})".format(roc\_auc))  
plt.xlabel("False Positive Rate")  
plt.ylabel("True Positive Rate")  
plt.title("ROC Curve")  
plt.legend(loc="lower right")  
plt.show()  
  
# 输出AUC面积  
# auc\_score = auc(fpr, tpr)  
print("AUC Score:", roc\_auc)  
  
from sklearn.linear\_model import SGDClassifier  
from sklearn.model\_selection import cross\_val\_score, cross\_val\_predict  
from sklearn.metrics import make\_scorer  
  
# 创建SGDClassifier模型  
sgd\_model = SGDClassifier()  
  
# 定义评估指标  
scoring = {  
 'accuracy': [],  
 'precision': [],  
 'recall': [],  
 'f1': []  
}  
# lst=['accuracy','precision','recall','f1']  
# print(lst[0])  
# 进行五折交叉验证  
for key,value in scoring.items():  
 scores = cross\_val\_score(sgd\_model, X\_test, y\_test, cv=5, scoring=key)  
 scoring[key].extend(scores)  
# 输出评估指标结果  
  
print("Cross Validation Results:")  
for key,value in scoring.items():  
 print(f'{key}:{np.array(value).mean()}')  
# print("Accuracy:", lst[0].mean())  
# print("Precision:", lst[1].mean())  
# print("Recall:", lst[2].mean())  
# print("F1-score:", lst[3].mean())



2、读取heart.csv 数据集，对数据进行预处理。

（1）使用两种分类模型中任意一种训练数据，输出训练集的混淆矩阵、准确率、精确率、召回率、F1-score，画精确率和召回率相对于阈值的函数图，绘制ROC曲线图，输出AUC面积。

（2）输入任意一个测试集的样本进行预测，输出预测值，并于真实值进行比较。

（3）使用五折交叉验证输出测试集的准确率、精确率、召回率、F1-score。

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score, cross\_validate  
from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_curve, auc, \  
 precision\_recall\_curve  
from sklearn.linear\_model import LogisticRegression  
from sklearn.preprocessing import StandardScaler  
from sklearn.model\_selection import KFold  
  
# 1. 加载数据集并进行预处理  
data = pd.read\_csv('heart.csv')  
# print(data)  
# 处理缺失值  
data.dropna(inplace=True)  
  
# 分割特征和标签  
X = data.drop('target', axis=1)  
y = data['target']  
  
# 数据标准化  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
# 分割数据集为训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
  
  
# 定义函数来训练模型并评估性能  
def evaluate\_model(model, X\_train, X\_test, y\_train, y\_test):  
 # 在训练集上训练模型  
 model.fit(X\_train, y\_train)  
  
 # 在训练集上进行预测  
 y\_pred\_train = model.predict(X\_train)  
  
 # 计算评估指标  
 cm = confusion\_matrix(y\_train, y\_pred\_train)  
 acc = accuracy\_score(y\_train, y\_pred\_train)  
 precision = precision\_score(y\_train, y\_pred\_train)  
 recall = recall\_score(y\_train, y\_pred\_train)  
 f1 = f1\_score(y\_train, y\_pred\_train)  
  
 # 绘制ROC曲线  
 y\_score = model.predict\_proba(X\_train)[:, 1]  
 precisions, recalls, thresholds = precision\_recall\_curve(y\_train, y\_score)  
 plt.plot(thresholds, precisions[:-1], label="Precision")  
 plt.plot(thresholds, recalls[:-1], label="Recall")  
 plt.xlabel("Threshold")  
 plt.legend(loc="upper left")  
 plt.title("Precision and Recall vs. Threshold")  
 plt.show()  
  
  
 fpr, tpr, thresholds = roc\_curve(y\_train, y\_score)  
 roc\_auc = auc(fpr, tpr)  
 plt.figure()  
 plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)  
 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('Receiver Operating Characteristic')  
 plt.legend(loc="lower right")  
 plt.show()  
  
 # 输出结果  
 print("Confusion Matrix:\n", cm)  
 print("Accuracy:", acc)  
 print("Precision:", precision)  
 print("Recall:", recall)  
 print("F1 Score:", f1)  
 print("AUC:", roc\_auc)  
  
  
# 2. 训练和评估模型  
# 逻辑回归模型  
logistic\_model = LogisticRegression()  
print("Logistic Regression Model:")  
evaluate\_model(logistic\_model, X\_train, X\_test, y\_train, y\_test)  
  
  
# 3. 预测一个样本并与真实值比较  
sample = X\_test[0].reshape(1, -1)  
print("\nSample Prediction:")  
print("Predicted:", logistic\_model.predict(sample))  
print("True Value:", y\_test.iloc[0])  
  
# 4. 使用五折交叉验证评估模型性能  
kf = KFold(n\_splits=5, shuffle=True, random\_state=42)  
scoring = ['accuracy', 'precision', 'recall', 'f1']  
logistic\_scores = cross\_validate(logistic\_model, X\_scaled, y, cv=kf, scoring=scoring)  
# print(logistic\_scores)  
print("\nCross-validated Logistic Regression Scores:")  
for score in scoring:  
 print(score.capitalize(), ":", np.mean(logistic\_scores['test\_' + score]))

