1、读取data\_multivar\_imbalance.txt数据，

（1）根据labels对样本前两列特征进行数据可视化

（2） 划分训练集和测试集，使用SVC模型训练数据，

      手动调参，核函数分别为linear, poly，sigmoid, rbf时，设置参数

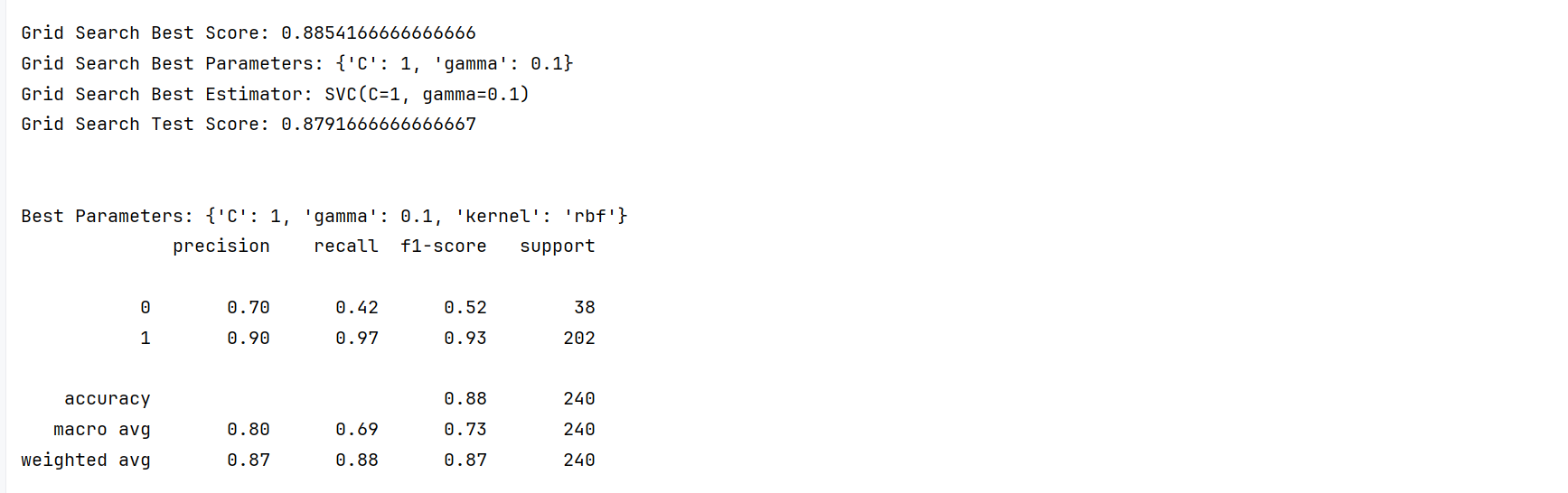
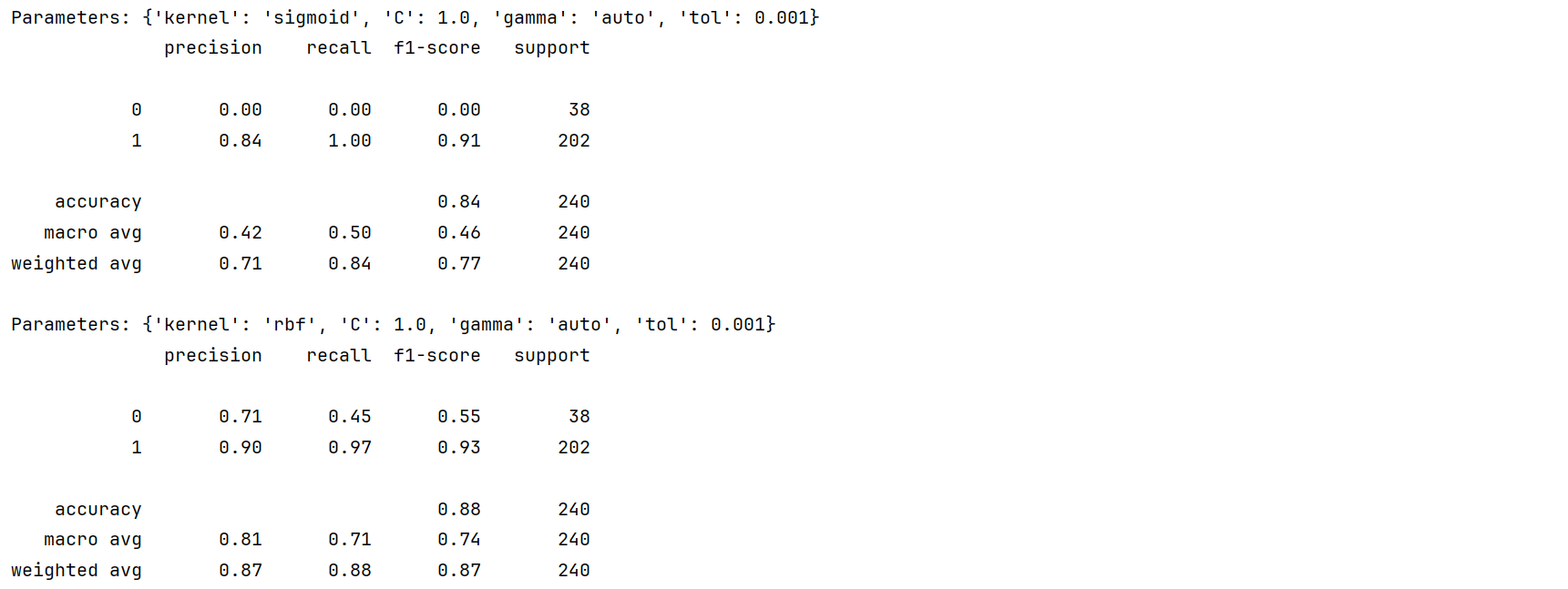
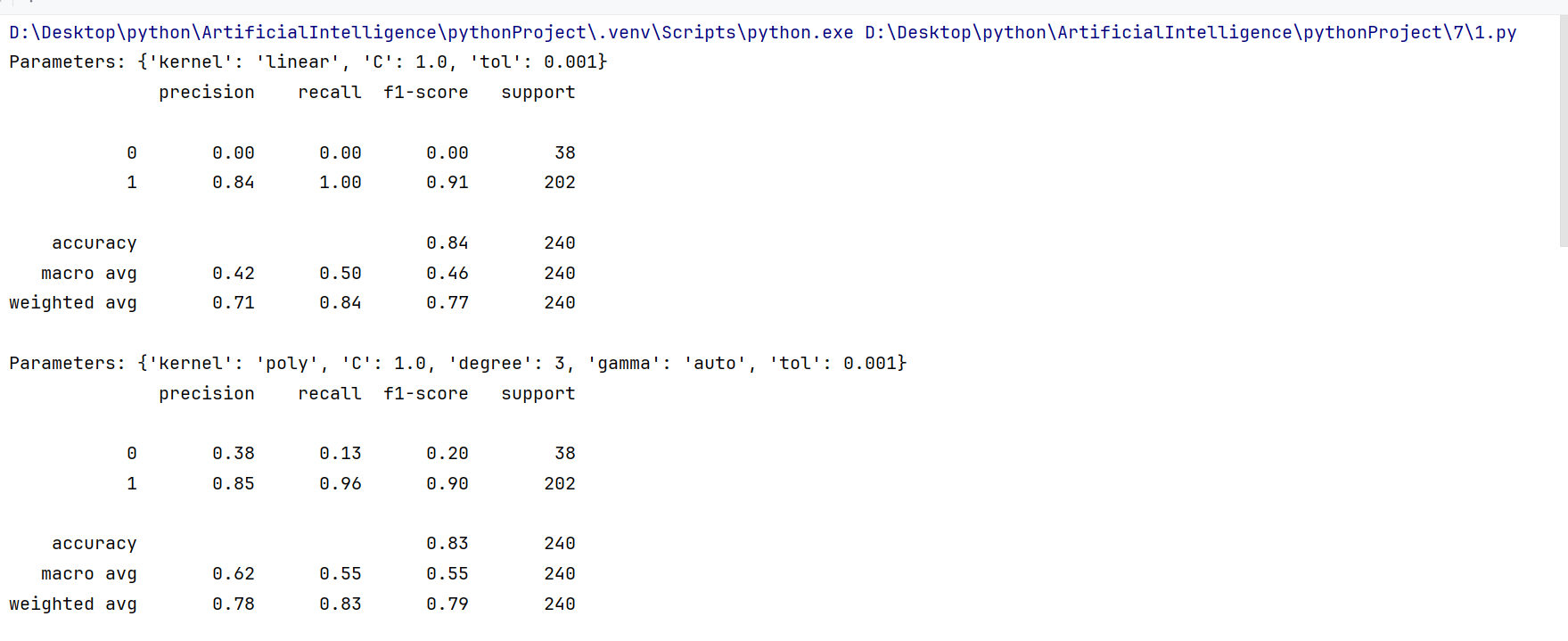
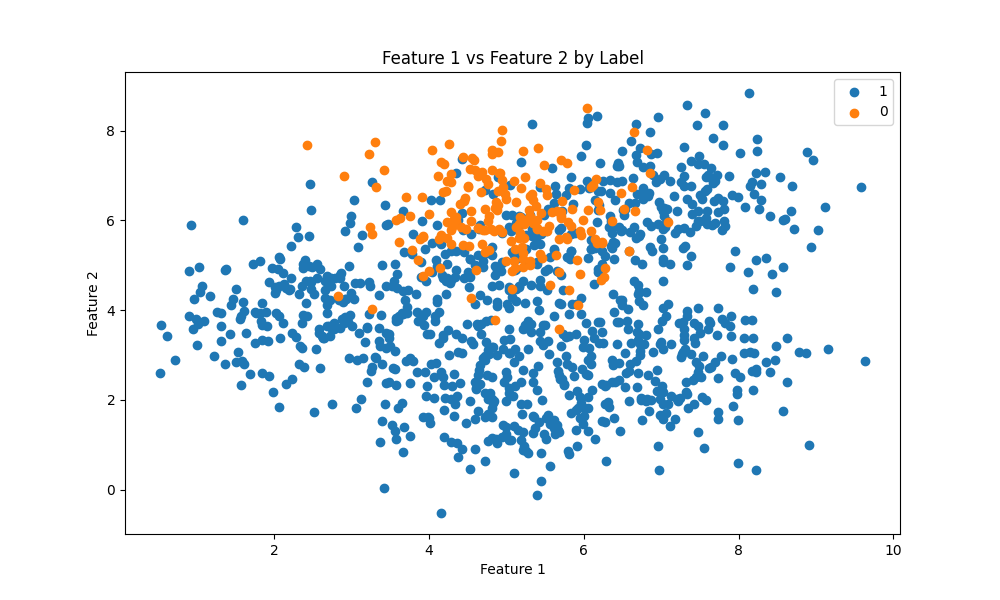
      输出不同参数下的测试集的性能报告

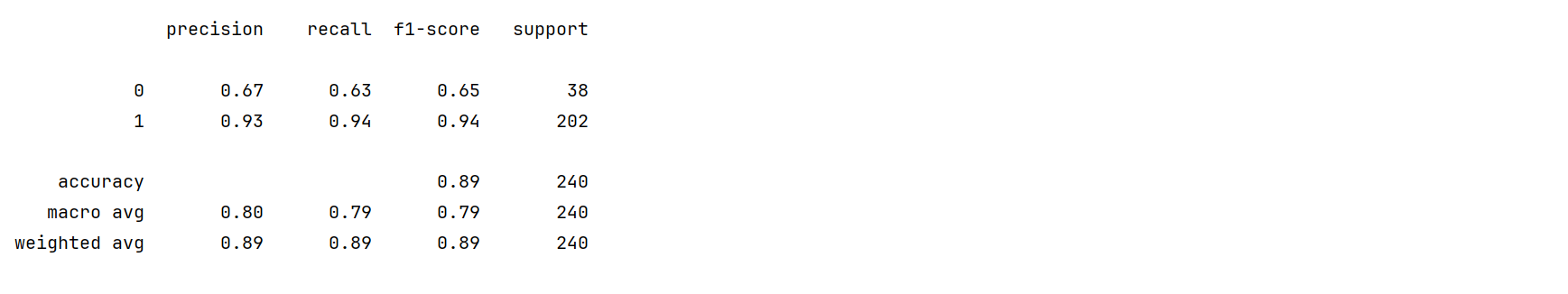
（3）使用网格调参，输出网格调参下的模型得分，最佳参数取值组合，最佳预测精度值，学习模型

（4）使用非网格调参，基于最高分数的参数集模型输出测试集的性能报告

（5）使用嵌套交叉验证机制下的网格参数寻优搜索，输出测试集的性能报告

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split, GridSearchCV, cross\_val\_score, cross\_val\_predict  
from sklearn.svm import SVC  
from sklearn.metrics import classification\_report  
import warnings  
warnings.filterwarnings("ignore")  
  
# 1. 读取data\_multivar\_imbalance.txt数据  
data = pd.read\_csv("data\_multivar\_imbalance.txt", header=None)  
  
# print(data)  
# 添加列名  
data.columns = ['feature1', 'feature2', 'labels']  
  
# 2. 根据labels对样本前两列特征进行数据可视化  
plt.figure(figsize=(10, 6))  
for label in data['labels'].unique():  
 subset = data[data['labels'] == label]  
 plt.scatter(subset['feature1'], subset['feature2'], label=label)  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.legend()  
plt.title('Feature 1 vs Feature 2 by Label')  
plt.show()  
  
# 3. 划分训练集和测试集  
X = data[['feature1', 'feature2']]  
y = data['labels']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# 设置SVC模型参数和核函数  
params = [  
 {'kernel':'linear','C':1.0,'tol':0.001},  
 {'kernel':'poly','C':1.0,'degree':3,'gamma':'auto','tol':0.001},  
 {'kernel':'sigmoid','C':1.0,'gamma':'auto','tol':0.001},  
 {'kernel': 'rbf', 'C': 1.0, 'gamma': 'auto', 'tol': 0.001}  
]  
  
# 4. 手动调参，输出不同参数下的测试集的性能报告  
for param in params:  
 svc = SVC(\*\*param)  
 svc.fit(X\_train, y\_train)  
 y\_pred = svc.predict(X\_test)  
 print("Parameters:", param)  
 print(classification\_report(y\_test, y\_pred))  
  
# 5. 使用网格调参  
param\_grid = {  
 'C': [0.1, 1, 10],  
 'gamma': [0.1, 1, 'scale', 'auto']  
}  
  
grid\_search = GridSearchCV(SVC(), param\_grid, cv=5)  
grid\_search.fit(X\_train, y\_train)  
  
# 输出网格调参下的模型得分，最佳参数取值组合，最佳预测精度值，学习模型  
print("Grid Search Best Score:", grid\_search.best\_score\_)  
print("Grid Search Best Parameters:", grid\_search.best\_params\_)  
print("Grid Search Best Estimator:", grid\_search.best\_estimator\_)  
print("Grid Search Test Score:", grid\_search.score(X\_test, y\_test))  
print('\n')  
# 6. 使用非网格调参，基于最高分数的参数集模型输出测试集的性能报告  
param\_grid2=[{'kernel':['linear'],'C':[1,10,50,600]},  
 {'kernel':['poly'],'degree':[2,3],'C':[1,10,50,600]},  
 {'kernel':['sigmoid'],'C':[1,10,50,600]},  
 {'kernel':['rbf'],'gamma':[0.1,1,10],'C':[1,10,50]}]  
grid\_search2 = GridSearchCV(SVC(), param\_grid2, cv=5)  
grid\_search2.fit(X\_train, y\_train)  
best\_params = grid\_search2.best\_params\_  
best\_svc = SVC(\*\*best\_params)  
best\_svc.fit(X\_train, y\_train)  
y\_pred = best\_svc.predict(X\_test)  
print("Best Parameters:", best\_params)  
print(classification\_report(y\_test, y\_pred))  
  
# 7. 使用嵌套交叉验证机制下的网格参数寻优搜索  
nested\_predict = cross\_val\_predict(grid\_search, X\_test, y\_test, cv=5)  
# print(nested\_predict.mean())  
print(classification\_report(y\_test,nested\_predict))  
# print("Nested Cross-Validation Score:", nested\_score.mean())



嵌套交叉验证机制下的网格参数寻优搜索，输出测试集的性能报告：

2、读取heart.csv数据

      （1） 划分训练集和测试集，使用SVM模型训练数据

          任选一种网格调参方式，输出调参下的模型得分，最佳参数取值组合，最佳预测精度值，学习模型

        （2）基于调参后的最优学习模型，输出测试集的性能报告

import pandas as pd  
from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.svm import SVC  
from sklearn.metrics import classification\_report  
  
# 1. 读取 heart.csv 数据集  
heart\_data = pd.read\_csv("../5/heart.csv")  
  
# 2. 划分训练集和测试集  
X = heart\_data.drop('target', axis=1)  
y = heart\_data['target']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# 3. 定义网格调参的参数范围  
param\_grid = {  
 'C': [0.1, 1, 10, 100],  
 'gamma': [0.1, 1, 10],  
 # 'kernel': ['linear', 'poly', 'rbf', 'sigmoid']  
}  
  
# 4. 使用网格调参  
grid\_search = GridSearchCV(SVC(), param\_grid, cv=5)  
grid\_search.fit(X\_train, y\_train)  
  
# 输出网格调参下的模型得分，最佳参数取值组合，最佳预测精度值，学习模型  
print("Grid Search Best Score:", grid\_search.best\_score\_)  
print("Grid Search Best Parameters:", grid\_search.best\_params\_)  
print("Grid Search Best Estimator:", grid\_search.best\_estimator\_)  
print("Grid Search Test Score:", grid\_search.score(X\_test, y\_test))  
  
# 5. 输出测试集的性能报告  
y\_pred = grid\_search.predict(X\_test)  
print("Test Set Performance Report:")  
print(classification\_report(y\_test, y\_pred))

