import numpy as np  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import classification\_report  
import matplotlib.pyplot as plt  
from matplotlib.colors import ListedColormap  
  
*# 读取数据集*df = pd.read\_csv("F:\DatasetLogisticRegression.txt", header=None, sep='\t')  
X = df.iloc[:, :-1].values *# 特征*y = df.iloc[:, -1].values *# 标签  
  
# 分离出测试集，每个类别10个样本*X\_train, X\_test, y\_train, y\_test = [], [], [], []  
for class\_value in np.unique(y):  
 X\_class = X[y == class\_value]  
 y\_class = y[y == class\_value]  
 X\_train\_class, X\_test\_class, y\_train\_class, y\_test\_class = train\_test\_split(  
 X\_class, y\_class, test\_size=10, random\_state=42, stratify=y\_class)  
 X\_train.append(X\_train\_class)  
 X\_test.append(X\_test\_class)  
 y\_train.append(y\_train\_class)  
 y\_test.append(y\_test\_class)  
  
X\_train = np.concatenate(X\_train)  
X\_test = np.concatenate(X\_test)  
y\_train = np.concatenate(y\_train)  
y\_test = np.concatenate(y\_test)  
  
*# 训练逻辑回归模型*logreg = LogisticRegression()  
logreg.fit(X\_train, y\_train)  
  
*# 预测测试集*y\_pred = logreg.predict(X\_test)  
  
*# 评价指标*print(classification\_report(y\_test, y\_pred))  
  
  
*# 可视化*def plot\_decision\_boundary(X, y, model):  
 h = .02 *# 网格中的步长* cmap\_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])  
 cmap\_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])  
  
 x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
 y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
 xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),  
 np.arange(y\_min, y\_max, h))  
 Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])  
 Z = Z.reshape(xx.shape)  
 plt.figure()  
 plt.pcolormesh(xx, yy, Z, cmap=cmap\_light)  
  
 *# 绘制训练点* plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap\_bold, edgecolor='k', s=20)  
 plt.xlim(xx.min(), xx.max())  
 plt.ylim(yy.min(), yy.max())  
 plt.title("Logistic Regression Decision Boundary")  
 plt.xlabel('Feature 1')  
 plt.ylabel('Feature 2')  
  
  
plot\_decision\_boundary(X\_train, y\_train, logreg)  
plt.scatter(X\_test[:, 0], X\_test[:, 1], color='yellow', edgecolor='k', s=100, label='test set')  
plt.legend()  
plt.show()