import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.datasets import load\_iris  
from sklearn.model\_selection import train\_test\_split  
  
# 1.读取数据  
import pandas as pd  
  
df = load\_iris()  
  
# 2.划分特征变量与目标变量  
X = df.data[:, :2]  
y = (df.target != 0) \* 1  
  
# 3.划分数据集与测试集  
from sklearn.model\_selection import train\_test\_split  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.15)  
  
# 4.模型搭建  
from sklearn.linear\_model import LogisticRegression  
  
model = LogisticRegression()  
model.fit(X\_train, y\_train)  
  
# 5.预测分类结果  
y\_train\_pred = model.predict(X\_train)  
print('训练集预测：', y\_train\_pred[:20])  
y\_test\_pred = model.predict(X\_test)  
print('测试集预测：', y\_test\_pred[:20])  
  
# 6.预测概率  
y\_train\_proba = model.predict\_proba(X\_train)  
print('训练集概率：', y\_train\_proba[:5])  
y\_test\_proba = model.predict\_proba(X\_test)  
print('测试集概率：', y\_test\_proba[:5])  
  
# 7.模型准确率  
from sklearn.metrics import accuracy\_score  
# accuracy = accuracy\_score(y\_test\_pred,y\_test)  
# 或者  
y\_train\_accuracy = model.score(X\_train, y\_train)  
y\_test\_accuracy = model.score(X\_test, y\_test)  
  
# 训练集精度  
print('训练集Accuracy: ', y\_train\_accuracy)  
# 测试集精度  
print('测试集Accuracy: ', y\_test\_accuracy)  
  
# 混淆矩阵  
from sklearn.metrics import confusion\_matrix  
m = confusion\_matrix(y\_test, y\_test\_pred)  
print('混淆矩阵:', m)  
  
# 计算不同阈值下的命中率（TPR）和假警报率（FPR）的值  
from sklearn.metrics import roc\_curve  
fpr, tpr, thres = roc\_curve(y\_test, y\_test\_pred)  
a = pd.DataFrame()  
a['阈值'] = list(thres)  
a['假报警率'] = list(fpr)  
a['命中率'] = list(tpr)  
print(a.head())  
  
# ROC曲线  
import matplotlib.pyplot as plt  
plt.figure()  
plt.plot(fpr, tpr)  
plt.title('ROC')  
plt.xlabel('FPR')  
plt.ylabel('TPR')  
plt.show()  
  
# KS曲线  
plt.figure()  
plt.plot(thres[1:], tpr[1:])  
plt.plot(thres[1:], fpr[1:])  
plt.plot(thres[1:], tpr[1:] - fpr[1:])  
plt.xlabel('threshold')  
plt.legend(['tpr', 'fpr', 'tpr-fpr'])  
plt.gca().invert\_xaxis()  
plt.show()  
  
# 可视化决策边界  
plt.figure()  
x1\_min, x1\_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5  
x2\_min, x2\_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5  
xx1, xx2 = np.meshgrid(np.linspace(x1\_min, x1\_max, 100), np.linspace(x2\_min, x2\_max, 100))  
Z = model.predict(np.c\_[xx1.ravel(), xx2.ravel()])  
Z = Z.reshape(xx1.shape)  
plt.contourf(xx1, xx2, Z, cmap=plt.cm.Spectral)  
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Spectral)  
plt.xlabel("Sepal length")  
plt.ylabel("Sepal width")  
plt.show()