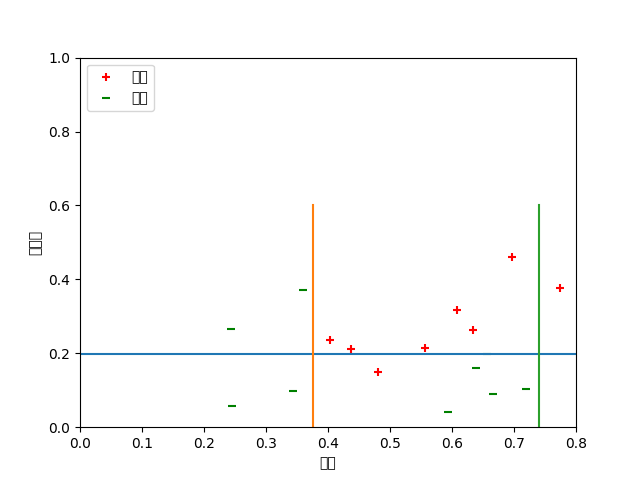


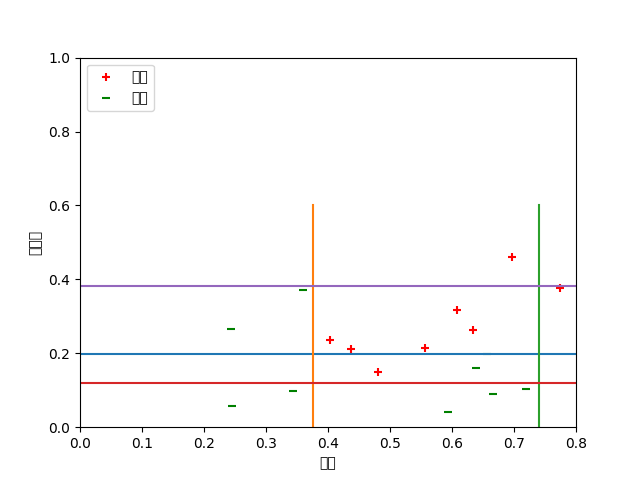
集成学习器（字典）： G3 = {0: {'alpha': 0.7702225204735745, 'stump': {'feature': 1, 'threshVal': 0.19875, 'inequal': 'it', 'err': 0.1764705882352941}}, 1: {'alpha': 0.7630281517475247, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.17857142857142855}}, 2: {'alpha': 0.5988515956561702, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'it', 'err': 0.23188405797101452}}}

准确率= 0.9411764705882353



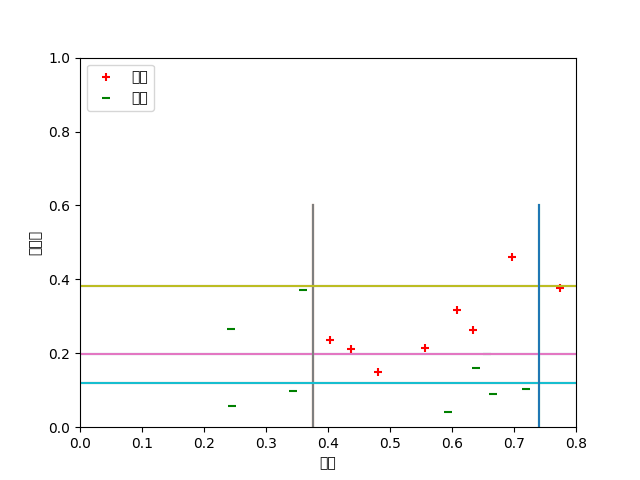
集成学习器（字典）： G5 = {0: {'alpha': 0.7702225204735745, 'stump': {'feature': 1, 'threshVal': 0.19875, 'inequal': 'it', 'err': 0.1764705882352941}}, 1: {'alpha': 0.7630281517475247, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.17857142857142855}}, 2: {'alpha': 0.5988515956561702, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'it', 'err': 0.23188405797101452}}, 3: {'alpha': 0.517116813427337, 'stump': {'feature': 1, 'threshVal': 0.12037500000000001, 'inequal': 'it', 'err': 0.2622641509433963}}, 4: {'alpha': 0.38449883125155576, 'stump': {'feature': 1, 'threshVal': 0.381625, 'inequal': 'it', 'err': 0.31669597186700765}}}

准确率= 0.9411764705882353



集成学习器（字典）： G11 = {0: {'alpha': 0.7702225204735745, 'stump': {'feature': 1, 'threshVal': 0.19875, 'inequal': 'it', 'err': 0.1764705882352941}}, 1: {'alpha': 0.7630281517475247, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.17857142857142855}}, 2: {'alpha': 0.5988515956561702, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'it', 'err': 0.23188405797101452}}, 3: {'alpha': 0.517116813427337, 'stump': {'feature': 1, 'threshVal': 0.12037500000000001, 'inequal': 'it', 'err': 0.2622641509433963}}, 4: {'alpha': 0.38449883125155576, 'stump': {'feature': 1, 'threshVal': 0.381625, 'inequal': 'it', 'err': 0.31669597186700765}}, 5: {'alpha': 0.47604085356392073, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.2784663666224749}}, 6: {'alpha': 0.44797704534424615, 'stump': {'feature': 1, 'threshVal': 0.19875, 'inequal': 'it', 'err': 0.28988264206782904}}, 7: {'alpha': 0.3050103325106539, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.352054483963029}}, 8: {'alpha': 0.38606192373470366, 'stump': {'feature': 1, 'threshVal': 0.381625, 'inequal': 'it', 'err': 0.31601985458121185}}, 9: {'alpha': 0.28200872281514566, 'stump': {'feature': 1, 'threshVal': 0.12037500000000001, 'inequal': 'it', 'err': 0.36261841086467406}}, 10: {'alpha': 0.22329255621950722, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'it', 'err': 0.39017299225192476}}}

准确率= 0.9411764705882353

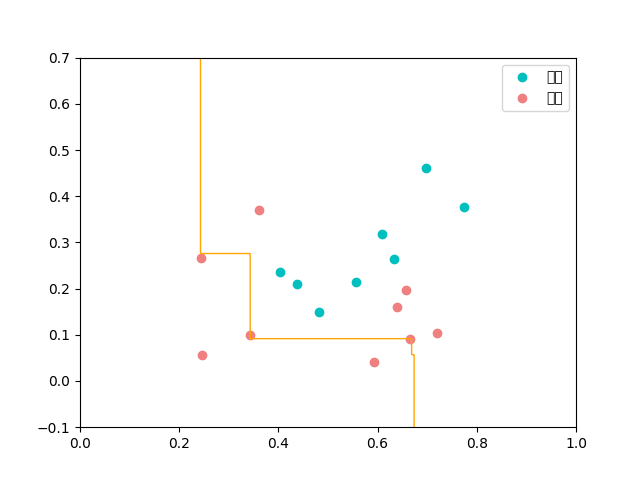


代码：

import base64  
import matplotlib  
import os  
import sys  
  
from matplotlib.\_pylab\_helpers import Gcf  
from matplotlib.backend\_bases import FigureManagerBase, ShowBase  
from matplotlib.backends.backend\_agg import FigureCanvasAgg  
from matplotlib.figure import Figure  
  
from datalore.display import debug, display, SHOW\_DEBUG\_INFO  
  
PY3 = sys.version\_info[0] >= 3  
  
index = int(os.getenv("PYCHARM\_MATPLOTLIB\_INDEX", 0))  
  
rcParams = matplotlib.rcParams  
  
  
class Show(ShowBase):  
 def \_\_call\_\_(self, \*\*kwargs):  
 debug("show() called with args %s" % kwargs)  
 managers = Gcf.get\_all\_fig\_managers()  
 if not managers:  
 debug("Error: Managers list in `Gcf.get\_all\_fig\_managers()` is empty")  
 return  
  
 for manager in managers:  
 manager.show(\*\*kwargs)  
  
 def mainloop(self):  
 pass  
  
  
show = Show()  
  
  
# from pyplot API  
def draw\_if\_interactive():  
 if matplotlib.is\_interactive():  
 figManager = Gcf.get\_active()  
 if figManager is not None:  
 figManager.canvas.show()  
 else:  
 debug("Error: Figure manager `Gcf.get\_active()` is None")  
  
  
# from pyplot API  
def new\_figure\_manager(num, \*args, \*\*kwargs):  
 FigureClass = kwargs.pop('FigureClass', Figure)  
 figure = FigureClass(\*args, \*\*kwargs)  
 return new\_figure\_manager\_given\_figure(num, figure)  
  
  
# from pyplot API  
def new\_figure\_manager\_given\_figure(num, figure):  
 canvas = FigureCanvasInterAgg(figure)  
 manager = FigureManagerInterAgg(canvas, num)  
 return manager  
  
  
# from pyplot API  
class FigureCanvasInterAgg(FigureCanvasAgg):  
 def \_\_init\_\_(self, figure):  
 FigureCanvasAgg.\_\_init\_\_(self, figure)  
  
 def show(self):  
 FigureCanvasAgg.draw(self)  
  
 if matplotlib.\_\_version\_\_ < '1.2':  
 buffer = self.tostring\_rgb(0, 0)  
 else:  
 buffer = self.tostring\_rgb()  
  
 if len(set(buffer)) <= 1:  
 # do not plot empty  
 debug("Error: Buffer FigureCanvasAgg.tostring\_rgb() is empty")  
 return  
  
 render = self.get\_renderer()  
 width = int(render.width)  
 debug("Image width: %d" % width)  
  
 is\_interactive = os.getenv("PYCHARM\_MATPLOTLIB\_INTERACTIVE", False)  
 if is\_interactive:  
 debug("Using interactive mode (Run with Python Console)")  
 debug("Plot index = %d" % index)  
 else:  
 debug("Using non-interactive mode (Run without Python Console)")  
 plot\_index = index if is\_interactive else -1  
 display(DisplayDataObject(plot\_index, width, buffer))  
  
 def draw(self):  
 FigureCanvasAgg.draw(self)  
 is\_interactive = os.getenv("PYCHARM\_MATPLOTLIB\_INTERACTIVE", False)  
 if is\_interactive and matplotlib.is\_interactive():  
 self.show()  
 else:  
 debug("Error: calling draw() in non-interactive mode won't show a plot. Try to 'Run with Python Console'")  
  
  
class FigureManagerInterAgg(FigureManagerBase):  
 def \_\_init\_\_(self, canvas, num):  
 FigureManagerBase.\_\_init\_\_(self, canvas, num)  
 global index  
 index += 1  
 self.canvas = canvas  
 self.\_num = num  
 self.\_shown = False  
  
 def show(self, \*\*kwargs):  
 self.canvas.show()  
 Gcf.destroy(self.\_num)  
  
  
class DisplayDataObject:  
 def \_\_init\_\_(self, plot\_index, width, image\_bytes):  
 self.plot\_index = plot\_index  
 self.image\_width = width  
 self.image\_bytes = image\_bytes  
  
 def \_repr\_display\_(self):  
 image\_bytes\_base64 = base64.b64encode(self.image\_bytes)  
 if PY3:  
 image\_bytes\_base64 = image\_bytes\_base64.decode()  
 body = {  
 'plot\_index': self.plot\_index,  
 'image\_width': self.image\_width,  
 'image\_base64': image\_bytes\_base64  
 }  
 return ('pycharm-plot-image', body)

截屏2022-12-16 09.50.00截屏2022-12-16 09.50.12

准确率0.7647058823529411

  
from matplotlib import pyplot as plt  
from sklearn.utils import resample  
  
def stumpClassify(X, dim, thresh\_val, thresh\_inequal):  
 ret\_array = np.ones((X.shape[0], 1))  
  
 if thresh\_inequal == 'lt':  
 ret\_array[X[:, dim] <= thresh\_val] = -1  
 else:  
 ret\_array[X[:, dim] > thresh\_val] = -1  
  
 return ret\_array  
  
#建立树桩  
def buildStump(X, y):  
 m, n = X.shape  
 best\_stump = {}  
 min\_error = 1  
 for dim in range(n):  
 x\_min = np.min(X[:, dim])  
 x\_max = np.max(X[:, dim])  
 # 这里第一次尝试使用排序后的点作为分割点，效果很差，因为那样会错过一些更好的分割点；  
 # 所以后来切割点改成将最大值和最小值之间分割成20等份。  
 # sorted\_x = np.sort(X[:, dim])  
 # split\_points = [(sorted\_x[i] + sorted\_x[i + 1]) / 2 for i in range(m - 1)]  
 split\_points = [(float(x\_max) - float(x\_min)) / 20 \* i + x\_min for i in range(20)]  
 for inequal in ['lt', 'gt']:  
 for thresh\_val in split\_points:  
 ret\_array = stumpClassify(X, dim, thresh\_val, inequal)  
 error = np.mean(ret\_array != y)  
 if error < min\_error:  
 best\_stump['dim'] = dim  
 best\_stump['thresh'] = thresh\_val  
 best\_stump['inequal'] = inequal  
 best\_stump['error'] = error  
 min\_error = error  
 return best\_stump  
  
def stumpBagging(X, y, nums=20):  
 stumps = []  
 seed = 16  
 for \_ in range(nums):  
 X\_, y\_ = resample(X, y, random\_state=seed) # sklearn 中自带的实现自助采样的方法  
 seed += 1  
 stumps.append(buildStump(X\_, y\_))  
 return stumps  
  
def stumpPredict(X, stumps):  
 ret\_arrays = np.ones((X.shape[0], len(stumps)))  
  
 for i, stump in enumerate(stumps):  
 ret\_arrays[:, [i]] = stumpClassify(X, stump['dim'], stump['thresh'], stump['inequal'])  
  
 return np.sign(np.sum(ret\_arrays, axis=1))  
  
#可视化  
def pltStumpBaggingDecisionBound(X\_, y\_, stumps):  
 pos = y\_ == 1  
 neg = y\_ == -1  
 x\_tmp = np.linspace(0, 1, 600)  
 y\_tmp = np.linspace(-0.1, 0.7, 600)  
  
 X\_tmp, Y\_tmp = np.meshgrid(x\_tmp, y\_tmp)  
 Z\_ = stumpPredict(np.c\_[X\_tmp.ravel(), Y\_tmp.ravel()], stumps).reshape(X\_tmp.shape)  
  
 plt.contour(X\_tmp, Y\_tmp, Z\_, [0], colors='orange', linewidths=1)  
  
 plt.scatter(X\_[pos, 0], X\_[pos, 1], label='好瓜', color='c')  
 plt.scatter(X\_[neg, 0], X\_[neg, 1], label='坏瓜', color='lightcoral')  
 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签  
 # plt.legend(loc='upper left')  
 plt.legend()  
 plt.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import numpy as np  
  
 # import pandas as pd  
 # data\_path = 'F:\\python\\dataset\\chapter8\\watermelon3\_0a\_Ch.txt'  
  
 # data = pd.read\_table(data\_path, delimiter=' ')  
  
 # X = data.iloc[:, :2].values  
 # y = data.iloc[:, 2].values  
 # XX = np.array(X)  
 # yy = np.array(y)  
 XX = [[0.697, 0.460],  
 [0.774, 0.376],  
 [0.634, 0.264],  
 [0.608, 0.318],  
 [0.556, 0.215],  
 [0.403, 0.237],  
 [0.481, 0.149],  
 [0.437, 0.211],  
 [0.666, 0.091],  
 [0.243, 0.267],  
 [0.245, 0.057],  
 [0.343, 0.099],  
 [0.639, 0.161],  
 [0.657, 0.198],  
 [0.360, 0.370],  
 [0.593, 0.042],  
 [0.719, 0.103]  
 ]  
 yy = [1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1]  
 X = np.array(XX)  
 y = np.array(yy)  
 stumps = stumpBagging(X, y, 21)  
  
 print(np.mean(stumpPredict(X, stumps) == y))  
 pltStumpBaggingDecisionBound(X, y, stumps)