```
import numpy as np
import matplotlib.pyplot as plt
from numpy import linspace
```

```
def getDataSet():
    dataSet = [
        [0.697, 0.460, '是'],
        [0.774, 0.376, '是'],
        [0.634, 0.264, '是'],
        [0.608, 0.318, '是'],
        [0.556, 0.215, '是'],
        [0.403, 0.237, '是'],
        [0.481, 0.149, '是'],
        [0.437, 0.211, '是'],
        [0.666, 0.091, '否'],
        [0.243, 0.267, '否'],
        [0.245, 0.057, '否'],
        [0.343, 0.099, '否'],
        [0.639, 0.161, '否'],
        [0.657, 0.198, '否'],
        [0.360, 0.370, '否'],
        [0.593, 0.042, '否'],
        [0.719, 0.103, '否']
    ]

    # 将是否为好瓜的字符替换为数字。替换是因为不想其他列的数值变成字符变量。
    for i in range(len(dataSet)):   # '是'换为1，'否'换为-1。
        if dataSet[i][-1] == '是':
            dataSet[i][-1] = 1
        else:
            dataSet[i][-1] = -1

    return np.array(dataSet)
```

```
def calErr(dataSet, feature, threshVal, inequal, D):
    """
    计算数据带权值的错误率。
    :param dataSet:     [密度，含糖量，好瓜]
    :param feature:     [密度，含糖量]
    :param threshVal:
    :param inequal:     'lt' or 'gt. (大于或小于）
    :param D:           数据的权重。错误分类的数据权重会大。
    :return:            错误率。
    """
    DFlatten = D.flatten()   # 变为一维
    errCnt = 0
    i = 0
    if inequal == 'lt':  #如果认为低于阈值为好瓜
        for data in dataSet:
            if (data[feature] <= threshVal and data[-1] == -1) or \
```

```
                (data[feature] > threshVal and data[-1] == 1):  #则错误判断 = 低于阈值且为坏瓜 + 高于阈值且为好瓜
                errCnt += 1 * DFlatten[i]  #该样本的权重作为错误率
            i += 1
    else:
        for data in dataSet:
            if (data[feature] >= threshVal and data[-1] == -1) or \
               (data[feature] < threshVal and data[-1] == 1):
                errCnt += 1 * DFlatten[i]
            i += 1
    return errCnt
```

```
def buildStump(dataSet, D):
    m, n = dataSet.shape
    bestErr = np.inf
    bestStump = {}
    numSteps = 16.0  # 每个特征迭代的步数
    for i in range(n-1):                    # 对第i个特征
        rangeMin = dataSet[:, i].min()
        rangeMax = dataSet[:, i].max()  # 每个属性列的最大最小值
        stepSize = (rangeMax - rangeMin) / numSteps  # 每一步的长度
        for j in range(m):                  # 对第j个数据
            threVal = rangeMin + float(j) * stepSize  # 每一步划分的阈值
            #threVal = dataSet[j][i]
            for inequal in ['lt', 'gt']:    # 对于大于或等于符号划分。
                err = calErr(dataSet, i, threVal, inequal, D)  # 错误率
                if err < bestErr:           # 如果错误更低，保存划分信息。
                    bestErr = err
                    bestStump["feature"] = i
                    bestStump["threshVal"] = threVal
                    bestStump["inequal"] = inequal
                    bestStump["err"] = err

    return bestStump
```

```
def predict(data, bestStump):
    if bestStump["inequal"] == 'lt':
        if data[bestStump["feature"]] <= bestStump["threshVal"]:
            return 1
        else:
            return -1
    else:
        if data[bestStump["feature"]] >= bestStump["threshVal"]:
            return 1
        else:
            return -1
```

```
def AdaBoost(dataSet, T):
    m, n = dataSet.shape
    D = np.ones((1, m)) / m                     # 初始化权重，每个样本的初始权重是相同的。
    classLabel = dataSet[:, -1].reshape(1, -1)   # 数据的类标签。
```

```
    G = {}       # 保存分类器的字典，

    for t in range(T):
        stump = buildStump(dataSet, D)            # 根据样本权重D建立一个决策树桩
        err = stump["err"]
        alpha = np.log((1 - err) / err) / 2       # 第t个分类器的权值
        # 更新训练数据集的权值分布
        pre = np.zeros((1, m))
        for i in range(m):
            pre[0][i] = predict(dataSet[i], stump)
        a = np.exp(-alpha * classLabel * pre)
        D = D * a / np.dot(D, a.T)

        G[t] = {}
        G[t]["alpha"] = alpha
        G[t]["stump"] = stump
    return G
```

```
def adaPredic(data, G):
    score = 0
    for key in G.keys():
        pre = predict(data, G[key]["stump"])  #每个基分类器的预测结果
        score += G[key]["alpha"] * pre        #加权结合后的集成预测结果
    flag = 0
    if score > 0:
        flag = 1
    else:
        flag = -1
    return flag
```

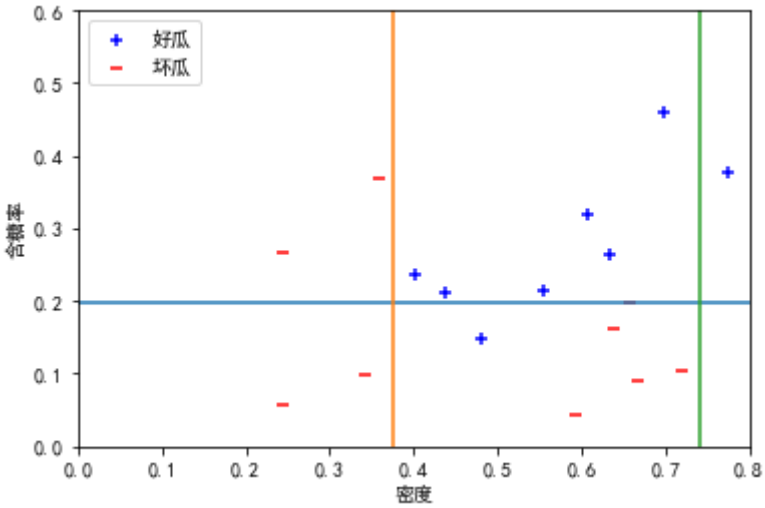
```
def calcAcc(dataSet, G):
    rightCnt = 0
    for data in dataSet:
        pre = adaPredic(data, G)
        if pre == data[-1]:
            rightCnt += 1
    return rightCnt / float(len(dataSet))
```

```
# 绘制数据集，clf为获得的集成学习器
def plotData(data, clf):
    X1, X2 = [], []
    Y1, Y2 = [], []
    datas=data
    labels=data[:,2]
    #print(np.argwhere(data==1))
    for data, label in zip(datas, labels):
        if label > 0:
            X1.append(data[0])
            Y1.append(data[1])
        else:
            X2.append(data[0])
            Y2.append(data[1])
```

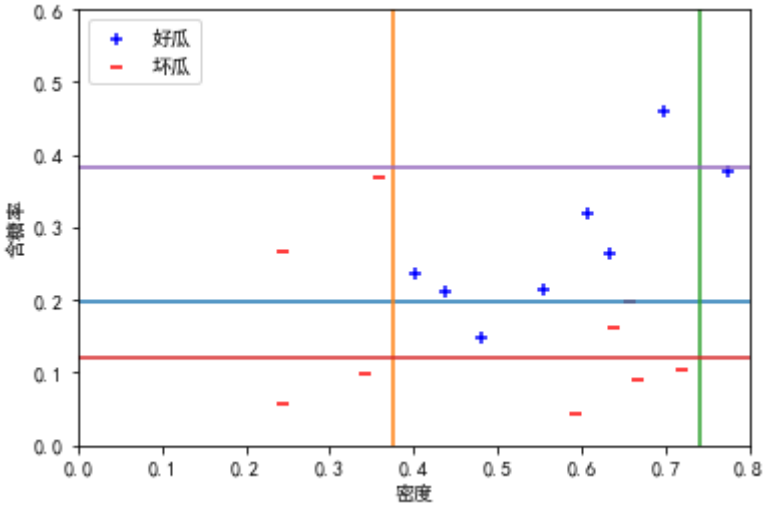
```
    x = linspace(0, 0.8, 100)
    y = linspace(0, 0.6, 100)

    for key in clf.keys():
        #print(clf[key]["stump"]["threshVal"])
        z = [clf[key]["stump"]["threshVal"]]*100
        if clf[key]["stump"]["feature"] == 0:
            plt.plot(z, y)
        else:
            plt.plot(x, z)

    plt.scatter(X1, Y1, marker='+', label='好瓜', color='b')
    plt.scatter(X2, Y2, marker='_', label='坏瓜', color='r')

    plt.xlabel('密度')
    plt.ylabel('含糖率')
    plt.xlim(0, 0.8)  # 设置x轴范围
    plt.ylim(0, 0.6)  # 设置y轴范围
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
    plt.legend(loc='upper left')
    plt.show()
```

```
dataSet = getDataSet()
for t in [3, 5, 11]:   # 学习器的数量
    G = AdaBoost(dataSet, t)
    print('集成学习器（字典）：',f"G{t} = {G}")
    print('准确率=',calcAcc(dataSet, G))
    #绘图函数
    plotData(dataSet,G)
```

```
集成学习器（字典）：  G3 = {0: {'alpha': 0.7702225204735745, 'stump': {'feature': 1,
'threshVal': 0.19875, 'inequal': 'gt', 'err': 0.1764705882352941}}, 1: {'alpha':
0.7630281517475247, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003,
'inequal': 'gt', 'err': 0.17857142857142855}}, 2: {'alpha': 0.5988515956561702,
'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'gt', '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': 'gt', 'err': 0.1764705882352941}}, 1: {'alpha': 0.7630281517475247, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'gt', 'err': 0.17857142857142855}}, 2: {'alpha': 0.5988515956561702, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'gt', 'err': 0.23188405797101452}}, 3: {'alpha': 0.517116813427337, 'stump': {'feature': 1, 'threshVal': 0.12037500000000001, 'inequal': 'gt', 'err': 0.2622641509433963}}, 4: {'alpha': 0.38449883125155576, 'stump': {'feature': 1, 'threshVal': 0.381625, 'inequal': 'gt', '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': 'gt', 'err': 0.1764705882352941}}, 1: {'alpha': 0.7630281517475247, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'gt', 'err': 0.17857142857142855}}, 2: {'alpha': 0.5988515956561702, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'gt', 'err': 0.23188405797101452}}, 3: {'alpha': 0.517116813427337, 'stump': {'feature': 1, 'threshVal': 0.12037500000000001, 'inequal': 'gt', 'err': 0.2622641509433963}}, 4: {'alpha': 0.38449883125155576, 'stump': {'feature': 1, 'threshVal': 0.381625, 'inequal': 'gt', 'err': 0.31669597186700765}}, 5: {'alpha': 0.47604085356392073, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'gt', 'err': 0.2784663666224749}}, 6: {'alpha': 0.5942588663532777, 'stump': {'feature': 0, 'threshVal': 0.574875, 'inequal': 'lt', 'err': 0.23352414290742707}}, 7: {'alpha': 0.36278381315927144, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'gt', 'err': 0.32616813391658545}}, 8: {'alpha': 0.5331663215215913, 'stump': {'feature': 1, 'threshVal': 0.381625, 'inequal': 'gt', 'err': 0.2561011394488732}}, 9: {'alpha': 0.4485878016499322, 'stump': {'feature': 1, 'threshVal': 0.19875, 'inequal': 'gt', 'err': 0.28963125739863155}}, 10: {'alpha': 0.5858601163372882, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'gt', 'err': 0.23654418489732032}}}
准确率= 1.0