```
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
import operator

featureDic = {
    '色泽': ['浅白', '青绿', '乌黑'],
    '根蒂': ['硬挺', '蜷缩', '稍蜷'],
    '敲声': ['沉闷', '浊响', '清脆'],
    '纹理': ['清晰', '模糊', '稍糊'],
    '脐部': ['凹陷', '平坦', '稍凹'],
    '触感': ['硬滑', '软粘']}


def getDataSet():
    """
    get watermelon data set 3.0.
    :return: 编码好的数据集以及特征的字典。
    """
    dataSet = [
        ['青绿', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', 0.697, 0.460, 1],
        ['乌黑', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑', 0.774, 0.376, 1],
        ['乌黑', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', 0.634, 0.264, 1],
        ['青绿', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑', 0.608, 0.318, 1],
        ['浅白', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', 0.556, 0.215, 1],
        ['青绿', '稍蜷', '浊响', '清晰', '稍凹', '软粘', 0.403, 0.237, 1],
        ['乌黑', '稍蜷', '浊响', '稍糊', '稍凹', '软粘', 0.481, 0.149, 1],
        ['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '硬滑', 0.437, 0.211, 1],
        ['乌黑', '稍蜷', '沉闷', '稍糊', '稍凹', '硬滑', 0.666, 0.091, 0],
        ['青绿', '硬挺', '清脆', '清晰', '平坦', '软粘', 0.243, 0.267, 0],
        ['浅白', '硬挺', '清脆', '模糊', '平坦', '硬滑', 0.245, 0.057, 0],
        ['浅白', '蜷缩', '浊响', '模糊', '平坦', '软粘', 0.343, 0.099, 0],
        ['青绿', '稍蜷', '浊响', '稍糊', '凹陷', '硬滑', 0.639, 0.161, 0],
        ['浅白', '稍蜷', '沉闷', '稍糊', '凹陷', '硬滑', 0.657, 0.198, 0],
        ['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '软粘', 0.360, 0.370, 0],
        ['浅白', '蜷缩', '浊响', '模糊', '平坦', '硬滑', 0.593, 0.042, 0],
        ['青绿', '蜷缩', '沉闷', '稍糊', '稍凹', '硬滑', 0.719, 0.103, 0]
    ]

    features = ['色泽', '根蒂', '敲声', '纹理', '脐部', '触感', '密度', '含糖量']
    # features = ['color', 'root', 'knocks', 'texture', 'navel', 'touch',
'density', 'sugar']

    # 每种特征的属性个数
    numList = []  # [3, 3, 3, 3, 3, 2]
    for i in range(len(features) - 2):
        numList.append(len(featureDic[features[i]]))

    dataSet = np.array(dataSet)
    return dataSet, features
```

```python
# data, classLabel, feature = getDataSet()
# print(data)
# print(classLabel)
# print(feature)

def cntProLap(dataSet, index, value, classLabel, N):

    extrData = dataSet[dataSet[:, -1] == classLabel]
    cnt = 0
    for data in extrData:
        if data[index] == value:
            cnt += 1
    return (cnt + 1) / (float(len(extrData)) + N)


def naiveBayesClassifier(dataSet, features):
    dict = {}
    for feature in features:
        index = features.index(feature)
        dict[feature] = {}
        if feature != '密度' and feature != '含糖量':
            featIList = featureDic[feature]
            for value in featIList:
                PisCond = cntProLap(dataSet, index, value, '1', len(featIList))
                pNoCond = cntProLap(dataSet, index, value, '0', len(featIList))
                dict[feature][value] = {}
                dict[feature][value]["是"] = PisCond
                dict[feature][value]["否"] = pNoCond
        else:
            for label in ['1', '0']:
                dataExtra = dataSet[dataSet[:, -1] == label]
                extr = dataExtra[:, index].astype("float64")
                aver = extr.mean()
                var = extr.var()

                labelStr = ""
                if label == '1':
                    labelStr = '是'
                else:
                    labelStr = '否'

                dict[feature][labelStr] = {}
                dict[feature][labelStr]["平均值"] = aver
                dict[feature][labelStr]["方差"] = var

    length = len(dataSet)
    classLabels = dataSet[:, -1].tolist()
    dict["好瓜"] = {}
    dict["好瓜"]['是'] = (classLabels.count('1') + 1) / (float(length) + 2)
    dict["好瓜"]['否'] = (classLabels.count('0') + 1) / (float(length) + 2)

    return dict


# # test naiveBayesClassifier(dataSet, features)
```

```
# dataSet, features = getDataSet()
# dic = naiveBayesClassifier(dataSet, features)
# print(dic)


def NormDist(mean, var, xi):
    return exp(-((float(xi) - mean) ** 2) / (2 * var)) / (sqrt(2 * pi * var))


def predict(data, features, bayesDis):
    pGood = bayesDis['好瓜']['是']
    pBad = bayesDis['好瓜']['否']
    for feature in features:
        index = features.index(feature)
        if feature != '密度' and feature != '含糖量':
            pGood *= bayesDis[feature][data[index]]['是']
            pBad *= bayesDis[feature][data[index]]['否']
        else:
            # NormDist(mean, var, xi)
            pGood *= NormDist(bayesDis[feature]['是']['平均值'],
                              bayesDis[feature]['是']['方差'],
                              data[index])
            pBad *= NormDist(bayesDis[feature]['否']['平均值'],
                             bayesDis[feature]['否']['方差'],
                             data[index])

    retClass = ""
    if pGood > pBad:
        retClass = "好瓜"
    else:
        retClass = "坏瓜"

    return pGood, pBad, retClass


def calcAccRate(dataSet, features, bayesDis):
    cnt = 0.0
    for data in dataSet:
        _, _, pre = predict(data, features, bayesDis)
        if (pre == '好瓜' and data[-1] == '1') \
            or (pre == '坏瓜' and data[-1] == '0'):
            cnt += 1

    return cnt / float(len(dataSet))


# test predict(data, features, bayesDis)
dataSet, features = getDataSet()
dic = naiveBayesClassifier(dataSet, features)
dic
```

```
{'色泽': {'浅白': {'是': 0.18181818181818182, '否': 0.4166666666666667},
  '青绿': {'是': 0.36363636363636365, '否': 0.3333333333333333},
```

```
 '乌黑': {'是': 0.45454545454545453, '否': 0.25}},
'根蒂': {'硬挺': {'是': 0.09090909090909091, '否': 0.25},
 '蜷缩': {'是': 0.5454545454545454, '否': 0.3333333333333333},
 '稍蜷': {'是': 0.36363636363636365, '否': 0.4166666666666667}},
'敲声': {'沉闷': {'是': 0.2727272727272727, '否': 0.3333333333333333},
 '浊响': {'是': 0.6363636363636364, '否': 0.4166666666666667},
 '清脆': {'是': 0.09090909090909091, '否': 0.25}},
'纹理': {'清晰': {'是': 0.7272727272727273, '否': 0.25},
 '模糊': {'是': 0.09090909090909091, '否': 0.3333333333333333},
 '稍糊': {'是': 0.18181818181818182, '否': 0.4166666666666667}},
'脐部': {'凹陷': {'是': 0.5454545454545454, '否': 0.25},
 '平坦': {'是': 0.09090909090909091, '否': 0.4166666666666667},
 '稍凹': {'是': 0.36363636363636365, '否': 0.3333333333333333}},
'触感': {'硬滑': {'是': 0.7, '否': 0.6363636363636364},
 '软粘': {'是': 0.3, '否': 0.36363636363636365}},
'密度': {'是': {'平均值': 0.57375, '方差': 0.014608437499999998},
 '否': {'平均值': 0.49611111111111117, '方差': 0.03370254320987655}},
'含糖量': {'是': {'平均值': 0.27875, '方差': 0.008912437500000002},
 '否': {'平均值': 0.1542222222222222, '方差': 0.01032861728395061 8}},
'好瓜': {'是': 0.47368421052631576, '否': 0.5263157894736842}}
```

```python
p1, p0, pre = predict(dataSet[0], features, dic)
print(f"p1 = {p1}")
print(f"p0 = {p0}")
print(f"pre = {pre}")
print("train data set acc = ", calcAccRate(dataSet, features, dic))
```

```
p1 = 0.02180124640594357
p0 = 4.915834021416594e-05
pre = 好瓜
train data set acc =  0.8235294117647058
```