

# Experiment2-1: Naive Bayes

Time: 2022/3/11

Location: Science\_Building\_119

Name: 易弘睿

Number: 20186103

## Part I: Review

朴素贝叶斯法是典型的生成学习方法。生成方法由训练数据学习联合概率分布 $P(X,Y)$ ，然后求得后验概率分布 $P(Y|X)$ 。具体来说，利用训练数据学习 $P(X|Y)$ 和 $P(Y)$ 的估计，得到联合概率分布： $P(X,Y)=P(X|Y)P(Y)$ 。计算公式如下：

The diagram illustrates the Naive Bayes formula with labels in Chinese:

- 后验概率** (Posterior Probability) points to  $P(Y|X)$ .
- 似然度** (Likelihood) points to  $P(X|Y)$ .
- 先验概率** (Prior Probability) points to  $P(Y)$ .
- 边际似然度** (Marginal Likelihood) points to  $P(X)$ .

$$P(Y|X) = \frac{P(X,Y)}{P(X)} = \frac{P(X|Y)P(Y)}{P(X)}$$

## Part II: Introduction

对初始代码的修改主要如下：

1. 对代码按照不同部分功能进行命名；
2. 对代码进行每行注释；
3. 对贝塞尔函数部分label和sample的定义区分开。

## Part III: Annotation

### 1. 数据库的导入

In [1]:

```
import pandas as pd # 导入pandas库
import numpy as np  # 导入numpy库
```

### 2. 数据的导入

In [2]:

```
data = pd.read_csv("西瓜.csv")
y = data['好瓜'].values.tolist() #将数组转化为列表
x = data[['色泽','根蒂','敲声','纹理','脐部','触感']].values.tolist()
data
```

Out[2]:

	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
0	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
1	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
3	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
4	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
5	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
6	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
7	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
8	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
9	青绿	硬挺	清脆	清晰	平坦	软粘	否
10	浅白	硬挺	清脆	模糊	平坦	硬滑	否
11	浅白	蜷缩	浊响	模糊	平坦	软粘	否
12	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
13	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
14	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
15	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
16	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

In [3]:

```
len(list(set(y)))
```

Out[3]:

2

### 3. 贝塞尔函数的定义

In [4]:

```
def naive_bayes(x, y, predict): # 设置函数
    unique_y = list(set(y)) # unique_y中存储两类，“是”表示好瓜，“否”表示坏瓜
    label_num = len(unique_y) # 长度为2（是、否）
    sample_num = len(x) # sample_num中存储总的的数据量
    dim = len(x[0]) # 标签个数
    joint_p = [1] * label_num # 初始化两种类别的概率都为1
    # 把所有的类别都过一遍，计算P(c)
    for (label_index, label) in enumerate(unique_y): # enumerate() 函数用于将一个
        p_c = len(data[data['好瓜'].isin([label])]) / sample_num # ‘好瓜’ 的概率（isin函数判
        for (feature_index, x_i) in enumerate(predict): # 把所有的类别都过一遍，计算
            tmp = data[data['好瓜'].isin([label])].values # 每个特征瓜个数（如果label为
            print(label, x_i)
            print(len([t for t in tmp[:, feature_index] if t == x_i]) / len(tmp)) # 计算每个特征的
            joint_p[label_index] *= len(
                [t for t in tmp[:, feature_index] if t == x_i]) / len(tmp) # 计算P(特征|类别)，比如P(
            joint_p[label_index] *= p_c # 几个标签出现最后满足好瓜的
    print(joint_p)

    # 输出最后的预测结果（是/否）
    tmp = joint_p[0]
    max_index = 0
    # 比较两个类别的概率，如果“是”的概率较大，那么最终判定这个西瓜为好瓜，否则为坏瓜
    for (i, p) in enumerate(joint_p):
        if tmp < p:
            tmp = p
            max_index = i

    return unique_y[max_index]
```

## 4. 数据的测试

In [5]:

```
out = naive_bayes(x, y, ["青绿", "蜷缩", "沉闷", "稍糊", "稍凹", "硬滑"])  
print(out)
```

```
否 青绿  
0.3333333333333333  
否 蜷缩  
0.3333333333333333  
否 沉闷  
0.3333333333333333  
否 稍糊  
0.4444444444444444  
否 稍凹  
0.3333333333333333  
否 硬滑  
0.6666666666666666  
是 青绿  
0.375  
是 蜷缩  
0.625  
是 沉闷  
0.25  
是 稍糊  
0.125  
是 稍凹  
0.375  
是 硬滑  
0.75  
[0.0019365770999757923, 0.000969381893382353]  
否
```