硕士第5周作业

编号	收入	学历	年龄	贷款
1	高	研究生	40-60	是
2	高	本科	>60	否
3	高	专科	20-40	是
4	中	研究生	40-60	是
5	中	本科	40-60	是
6	中	专科	20-40	是
7	中	研究生	>60	是
8	低	本科	>60	否
9	低	本科	20-40	是
10	低	专科	40-60	否
11	低	专科	>60	否

计算收入中等，20-40本科生是否应该贷款
（计算过程+代码）

记：
收入：高，中，低 > 2，1，0
学历：研究生，本科，专科 > 2，1，0
年龄：>60，40-60，20-40 > 2，1，0
有：

```
X = np.array([
    [2,2,1],
    [2,1,2],
    [2,0,0],
    [1,2,1],
    [1,1,1],
    [1,0,0],
    [1,2,2],
    [0,1,2],
    [0,1,0],
    [0,0,1],
    [0,0,2]
])
y = np.array([1,0,1,1,1,1,1,0,1,0,0])
```

所问问题收入中等，20-40本科生为[1,1,0]
因为收入中等未贷款的样本数为0，以及年龄20-40未贷款的样本数也为0，因此需要使用拉普拉斯平滑

手动计算

$$
\begin{aligned}
&p(y=1|x^{(1)}=1,x^{(2)}=1,x^{(3)}=0)\\
&=p(y=1)\times p(x^{(1)}=1|y=1)\times p(x^{(2)}=1|y=1)\times p(x^{(3)}=0|y=1)\\
&=\frac{7}{11}\times\frac{4+1}{7+3}\times\frac{2+1}{7+3}\times\frac{3+1}{7+3}\\
&=\frac{21}{550}
\end{aligned}
$$

$$
\begin{aligned}
&p(y=0|x^{(1)}=1,x^{(2)}=1,x^{(3)}=0)\\
&=p(y=0)\times p(x^{(1)}=1|y=0)\times p(x^{(2)}=1|y=0)\times p(x^{(3)}=0|y=0)\\
&=\frac{4}{11}\times\frac{0+1}{4+3}\times\frac{2+1}{4+3}\times\frac{0+1}{4+3}\\
&=\frac{12}{3773}
\end{aligned}
$$

$$
P(y'=1)=\frac{\frac{21}{550}}{\frac{21}{550}+\frac{12}{3773}}=\frac{79233}{85833}=0.9231
$$

$$
P(y'=0)=\frac{\frac{12}{3773}}{\frac{21}{550}+\frac{12}{3773}}=\frac{6600}{85833}=0.0769
$$

即贷款的概率为0.9231，不贷款的概率为0.07689

因此收入中等，20-40本科生应贷款

代码

```
import numpy as np
X = np.array([
    [2,2,1],
    [2,1,2],
    [2,0,0],
    [1,2,1],
    [1,1,1],
    [1,0,0],
    [1,2,2],
    [0,1,2],
    [0,1,0],
    [0,0,1],
    [0,0,2]
])

y = np.array([1,0,1,1,1,1,1,0,1,0,0])

# 朴素贝叶斯类，面向离散特征
class NaiveBayes:
    def __init__(self):
        self.prior = {}
        self.prior_num = {}
        self.posterior = {}

    # 训练模型
```

```
    def fit(self, X, y):

        # 计算先验概率  数据结构形式：{0: 0.5, 1: 0.5}
        labels = list(set(y))
        self.prior = {label: 0 for label in labels}
        self.prior_num = {label: 0 for label in labels}
        for value in y:
            self.prior[value] +=1.0/ (len(y)) #累计计算先验概率
            self.prior_num[value] += 1 #累计不同分类的样本个数，备用

        self.prior_num[0] += 3
        self.prior_num[1] += 3
        # print(self.prior_num)

        # 计算后验概率  数据结构形式：{0: [{0: 0.75, 1: 0.25}, {0: 0.5, 1: 0.5}......}
        self.posterior = {label: [] for label in labels}
        for label in labels:
            for _ in range(X.shape[-1]):
                self.posterior[label].append({}) # 为每个类别，初始化一个空字典
        for item, label in zip(X, y):
            prior_num_y = self.prior_num[label]
            for i, val in enumerate(item):
                if val in self.posterior[label][i]:
                    self.posterior[label][i][val] += 1.0/prior_num_y  #已存在该特征值，则进行累计计算
                else:
                    self.posterior[label][i][val] = 1.0/prior_num_y  #不存在该特征值，则初始化

        print("先验概率:",self.prior)
        print("后验概率:",self.posterior)

    # 通过已知的概率，对样本做预测
    def predict_single(self, X_test):
        results = {}
        # 通过朴素贝叶斯公式来计算
        for label,prior_val in self.prior.items():
            results[label] = prior_val
            for i, post_val in enumerate(X_test):
                if(post_val in self.posterior[label][i]):
                    results[label] *= (self.posterior[label][i][post_val] + 1 / self.prior_num[label])
```

```
                else:
                    results[label] *= 1 / self.prior_num[label]
        denominator = np.sum(list(results.values()))
        # 返回不同分类的概率
        for label in results.keys():
            results[label] = results[label]/denominator
        return results
nb = NaiveBayes()
nb.fit(X,y)
test_sample = [1,1,0]
print("测试结果:", nb.predict_single(test_sample))
```

```
先验概率: {0: 0.36363636363636365, 1: 0.6363636363636365}
后验概率: {0: [{2: 0.14285714285714285, 0: 0.42857142857142855}, {1: 0.2857142857142857, 0: 0.2857142857142857}, {2: 0.42857142857142855, 1: 0.14285714285714285}], 1: [{2: 0.2, 1: 0.4, 0: 0.1}, {2: 0.30000000000000004, 0: 0.2, 1: 0.2}, {1: 0.30000000000000004, 0: 0.30000000000000004, 2: 0.1}]}
测试结果: {0: 0.07689350249903878, 1: 0.9231064975009613}
```

```
from sklearn.naive_bayes import CategoricalNB
import numpy as np

X = np.array([
    [2,2,1],
    [2,1,2],
    [2,0,0],
    [1,2,1],
    [1,1,1],
    [1,0,0],
    [1,2,2],
    [0,1,2],
    [0,1,0],
    [0,0,1],
    [0,0,2]
])

y = np.array([1,0,1,1,1,1,1,0,1,0,0])


mnb = CategoricalNB(alpha = 1)
mnb.fit(X,y)

test_sample = np.array([[1,1,0]])
print("测试结果:", mnb.predict_proba(test_sample))
```

```
print('测试结果：' + mlb.predict_proba(test_sample))
```

```
测试结果：[[0.0768935 0.9231065]]
```