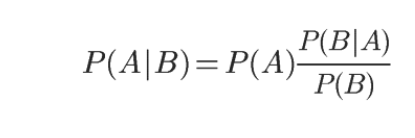
**AI课上代码实现 第七次试验 硬件一班 王倩倩 171491121**

1. **朴素贝叶斯——python实现**
2. **简单了解朴素贝叶斯**



我们把P(A)称为"先验概率"（Prior probability），即在B事件发生之前，我们对A事件概率的一个判断。

P(A|B)称为"后验概率"（Posterior probability），即在B事件发生之后，我们对A事件概率的重新评估。

P(B|A)/P(B)称为"可能性函数"（Likelyhood），这是一个调整因子，使得预估概率更接近真实概率。

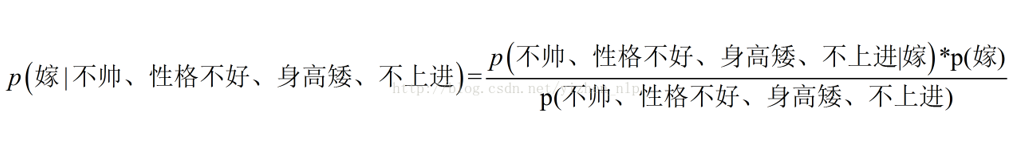
朴素贝叶斯“朴素”在于：利用贝叶斯定理求解联合概率P(XY)时，需要计算条件概率P(X|Y)。在计算P(X|Y)时，朴素贝叶斯做了一个很强的条件独立假设（当Y确定时，X的各个分量取值之间相互独立），即P(X1=x1,X2=x2,…Xj=xj|Y=yk) = P(X1=x1|Y=yk)P(X2=x2|Y=yk)…\*P(Xj=xj|Y=yk)。

一个例子如下：



问题是，如果一对男女朋友，男生想女生求婚，男生的四个特点分别是不帅，性格不好，身高矮，不上进，请你判断一下女生是嫁还是不嫁？

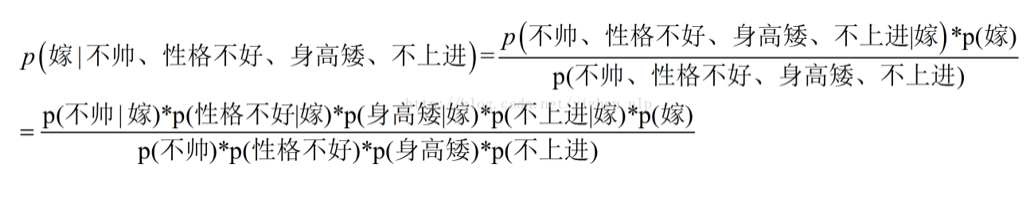
这是一个典型的分类问题，转为数学问题就是比较p(嫁|(不帅、性格不好、身高矮、不上进))与p(不嫁|(不帅、性格不好、身高矮、不上进))的概率，谁的概率大，就能给出嫁或者不嫁的答案！



只要求得p(不帅、性格不好、身高矮、不上进|嫁)、p（不帅、性格不好、身高矮、不上进)、p(嫁)即可

p(不帅、性格不好、身高矮、不上进|嫁) = p(不帅|嫁)\*p(性格不好|嫁)\*p(身高矮|嫁)\*p(不上进|嫁)，而这是因为特征之间相互独立，朴素贝叶斯法对条件概率分布做了条件独立性的假设。

上述公式就变为：



我们一个一个进行计算

p(嫁)=？

首先我们整理训练数据中，嫁的样本数如下：



则 p(嫁) = 6/12（总样本数） = 1/2

p(不帅|嫁)=？统计满足样本数如下：



则p(不帅|嫁) = 3/6 = 1/2 在嫁的条件下，看不帅有多少

p(性格不好|嫁)= ？统计满足样本数如下：



则p(性格不好|嫁)= 1/6

p（矮|嫁） = ?统计满足样本数如下：



则p(矮|嫁) = 1/6

p(不上进|嫁) = ?统计满足样本数如下：



则p(不上进|嫁) = 1/6

下面开始求分母，p(不帅)，p（性格不好），p（矮），p（不上进）

统计样本如下：



不帅统计如上红色所示，占4个，那么p（不帅） = 4/12 = 1/3



性格不好统计如上红色所示，占4个，那么p（性格不好） = 4/12 = 1/3

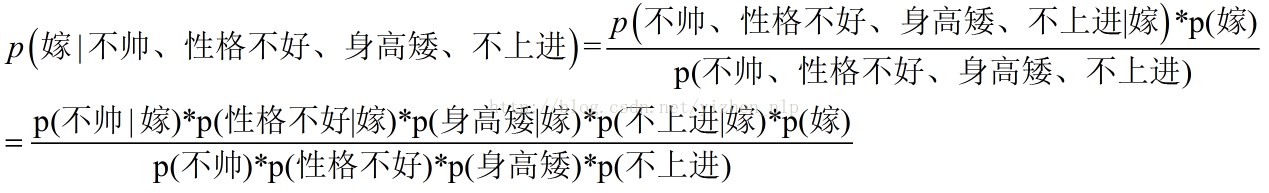


身高矮统计如上红色所示，占7个，那么p（身高矮） = 7/12



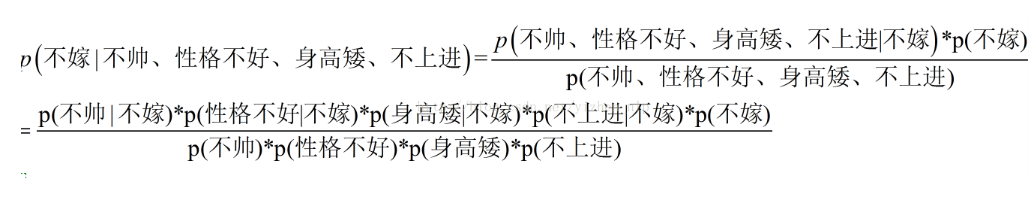
不上进统计如上红色所示，占4个，那么p（不上进） = 4/12 = 1/3

到这里，要求p(不帅、性格不好、身高矮、不上进|嫁)的所需项全部求出来了，下面我带入进去即可，



= (1/2\*1/6\*1/6\*1/6\*1/2)/(1/3\*1/3\*7/12\*1/3)

下面我们根据同样的方法来求p(不嫁|不帅，性格不好，身高矮，不上进)



p (不嫁|不帅、性格不好、身高矮、不上进) = ((1/6\*1/2\*1\*1/2)\*1/2)/(1/3\*1/3\*7/12\*1/3)

很显然(1/6\*1/2\*1\*1/2) > (1/2\*1/6\*1/6\*1/6\*1/2)

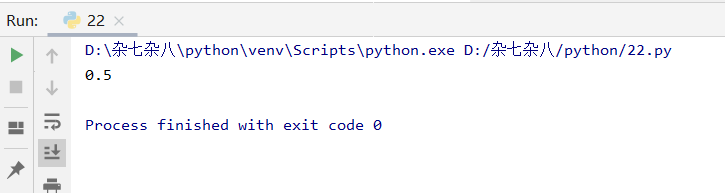
于是有p (不嫁|不帅、性格不好、身高矮、不上进)>p (嫁|不帅、性格不好、身高矮、不上进)

所以我们根据朴素贝叶斯算法可以给这个女生答案，是不嫁。

1. 代码实现

**import** random  
**def** create\_Data():  
 datasets=[  
 [**'帅'**,**'不好'**,**'矮'**,**'不上进'**,**'不嫁'**],  
 [**'不帅'**,**'好'**,**'矮'**,**'上进'**,**'不嫁'**],  
 [**'帅'**,**'好'**,**'矮'**,**'上进'**,**'嫁'**],  
 [**'不帅'**,**'好'**,**'高'**,**'上进'**,**'嫁'**],  
 [**'帅'**,**'不好'**,**'矮'**,**'上进'**,**'不嫁'**],  
 [**'不帅'**,**'不好'**,**'矮'**,**'不上进'**,**'不嫁'**],  
 [**'帅'**,**'好'**,**'高'**,**'不上进'**,**'嫁'**],  
 [**'不帅'**,**'好'**,**'高'**,**'上进'**,**'嫁'**],  
 [**'帅'**,**'好'**,**'高'**,**'上进'**,**'嫁'**],  
 [**'不帅'**,**'不好'**,**'高'**,**'上进'**,**'嫁'**],  
 [**'帅'**,**'好'**,**'矮'**,**'不上进'**,**'不嫁'**],  
 [**'帅'**,**'好'**,**'矮'**,**'不上进'**,**'不嫁'**]  
 ]  
 **return** datasets  
*# 计算函数***def** compute\_threeProb(datasets,c1,c2,c3,c4,c5):*# 传入训练数据集，需要判断的数据（帅不帅，性格好不好，高还是矮，上不上进，嫁不嫁）* C5\_count=0 *#满足c5条件的数据条数* Result\_count=0 *#满足c5条件下且满足需要判断的四个特征的个数* p3\_count=0 *#计算样本中符合要判断的四个特征的个数* Allcount=len(datasets) *#数据样本条数* **for** dataset **in** datasets:  
 *#满足c5条件的数据条数* **if** dataset[4]==c5:  
 C5\_count+=1  
 *#该类别下满足需要判断的四个特征的个数* **if** dataset[0]==c1 **and** dataset[1]== c2 **and** dataset[2]==c3 **and** dataset[3]==c4:  
 Result\_count+=1  
 *#计算样本中符合要判断的四个特征的个数* **if** dataset[0] == c1 **and** dataset[1] == c2 **and** dataset[2] == c3 **and** dataset[3] == c4:  
 p3\_count+=1  
  
 p1=(C5\_count/Allcount)\*(Result\_count/C5\_count) *#类似 P(不帅，性格不好，不高，不上进|嫁)* p2=C5\_count/Allcount *#类似 P(嫁)* p3=p3\_count/Allcount *#类似 P(不帅，性格不好，不高，不上进）* **if** p3!=0:  
 prob\_marriage=p1\*p2/p3  
 print(prob\_marriage)  
 **return** prob\_marriage  
 **else**:  
 print(**"不存在！"**)  
 **return** 0  
 *# 调用*datasets=create\_Data()  
  
rel = compute\_threeProb(datasets, c1=**'不帅'**, c2=**'不好'**, c3=**'矮'**, c4=**'不上进'**, c5=**'不嫁'**)

三、运行结果



1. 朴素贝叶斯的另一个应用——文本分类器

 以在线社区留言为例。为了不影响社区的发展，屏蔽侮辱性的言论，所以要构建一个快速过滤器，如果某条留言使用了负面或者侮辱性的语言，那么就将该留言标志为内容不当。过滤这类内容是一个很常见的需求。对此问题建立两个类型：侮辱类和非侮辱类，使用1和0分别表示。

**代码实现：**

**import** numpy **as** np  
**from** functools **import** reduce  
  
**"""  
函数说明:创建实验样本  
  
Parameters:  
 无  
Returns:  
 postingList - 实验样本切分的词条  
 classVec - 类别标签向量  
  
"""  
def** loadDataSet():  
 postingList=[[**'my'**, **'dog'**, **'has'**, **'flea'**, **'problems'**, **'help'**, **'please'**], *#切分的词条* [**'maybe'**, **'not'**, **'take'**, **'him'**, **'to'**, **'dog'**, **'park'**, **'stupid'**],  
 [**'my'**, **'dalmation'**, **'is'**, **'so'**, **'cute'**, **'I'**, **'love'**, **'him'**],  
 [**'stop'**, **'posting'**, **'stupid'**, **'worthless'**, **'garbage'**],  
 [**'mr'**, **'licks'**, **'ate'**, **'my'**, **'steak'**, **'how'**, **'to'**, **'stop'**, **'him'**],  
 [**'quit'**, **'buying'**, **'worthless'**, **'dog'**, **'food'**, **'stupid'**]]  
 classVec = [0,1,0,1,0,1] *#类别标签向量，1代表侮辱性词汇，0代表不是* **return** postingList,classVec *#返回实验样本切分的词条和类别标签向量***"""  
函数说明:将切分的实验样本词条整理成不重复的词条列表，也就是词汇表  
  
Parameters:  
 dataSet - 整理的样本数据集  
Returns:  
 vocabSet - 返回不重复的词条列表，也就是词汇表  
  
"""  
def** createVocabList(dataSet):  
 vocabSet = set([]) *#创建一个空的不重复列表* **for** document **in** dataSet:  
 vocabSet = vocabSet | set(document) *#取并集* **return** list(vocabSet)  
  
**"""  
函数说明:根据vocabList词汇表，将inputSet向量化，向量的每个元素为1或0  
  
Parameters:  
 vocabList - createVocabList返回的列表  
 inputSet - 切分的词条列表  
Returns:  
 returnVec - 文档向量,词集模型  
  
"""  
def** setOfWords2Vec(vocabList, inputSet):  
 returnVec = [0] \* len(vocabList) *#创建一个其中所含元素都为0的向量* **for** word **in** inputSet: *#遍历每个词条* **if** word **in** vocabList: *#如果词条存在于词汇表中，则置1* returnVec[vocabList.index(word)] = 1  
 **else**: print(**"the word: %s is not in my Vocabulary!"** % word)  
 **return** returnVec *#返回文档向量***"""  
函数说明:朴素贝叶斯分类器训练函数  
  
Parameters:  
 trainMatrix - 训练文档矩阵，即setOfWords2Vec返回的returnVec构成的矩阵  
 trainCategory - 训练类别标签向量，即loadDataSet返回的classVec  
Returns:  
 p0Vect - 侮辱类的条件概率数组  
 p1Vect - 非侮辱类的条件概率数组  
 pAbusive - 文档属于侮辱类的概率  
  
"""  
def** trainNB0(trainMatrix,trainCategory):  
 numTrainDocs = len(trainMatrix) *#计算训练的文档数目* numWords = len(trainMatrix[0]) *#计算每篇文档的词条数* pAbusive = sum(trainCategory)/float(numTrainDocs) *#文档属于侮辱类的概率* p0Num = np.zeros(numWords); p1Num = np.zeros(numWords) *#创建numpy.zeros数组,* p0Denom = 0.0; p1Denom = 0.0 *#分母初始化为0.0* **for** i **in** range(numTrainDocs):  
 **if** trainCategory[i] == 1: *#统计属于侮辱类的条件概率所需的数据，即P(w0|1),P(w1|1),P(w2|1)···* p1Num += trainMatrix[i]  
 p1Denom += sum(trainMatrix[i])  
 **else**: *#统计属于非侮辱类的条件概率所需的数据，即P(w0|0),P(w1|0),P(w2|0)···* p0Num += trainMatrix[i]  
 p0Denom += sum(trainMatrix[i])  
 p1Vect = p1Num/p1Denom *#相除* p0Vect = p0Num/p0Denom  
 **return** p0Vect,p1Vect,pAbusive *#返回属于侮辱类的条件概率数组，属于非侮辱类的条件概率数组，文档属于侮辱类的概率***"""  
函数说明:朴素贝叶斯分类器分类函数  
  
Parameters:  
 vec2Classify - 待分类的词条数组  
 p0Vec - 侮辱类的条件概率数组  
 p1Vec -非侮辱类的条件概率数组  
 pClass1 - 文档属于侮辱类的概率  
Returns:  
 0 - 属于非侮辱类  
 1 - 属于侮辱类  
  
"""  
def** classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):  
 p1 = reduce(**lambda** x,y:x\*y, vec2Classify \* p1Vec) \* pClass1 *#对应元素相乘* p0 = reduce(**lambda** x,y:x\*y, vec2Classify \* p0Vec) \* (1.0 - pClass1)  
 print(**'p0:'**,p0)  
 print(**'p1:'**,p1)  
 **if** p1 > p0:  
 **return** 1  
 **else**:  
 **return** 0  
  
**"""  
函数说明:测试朴素贝叶斯分类器  
  
Parameters:  
 无  
Returns:  
 无  
  
"""  
def** testingNB():  
 listOPosts,listClasses = loadDataSet() *#创建实验样本* myVocabList = createVocabList(listOPosts) *#创建词汇表* trainMat=[]  
 **for** postinDoc **in** listOPosts:  
 trainMat.append(setOfWords2Vec(myVocabList, postinDoc)) *#将实验样本向量化* p0V,p1V,pAb = trainNB0(np.array(trainMat),np.array(listClasses)) *#训练朴素贝叶斯分类器* testEntry = [**'love'**, **'my'**, **'dalmation'**] *#测试样本1* thisDoc = np.array(setOfWords2Vec(myVocabList, testEntry)) *#测试样本向量化* **if** classifyNB(thisDoc,p0V,p1V,pAb):  
 print(testEntry,**'属于侮辱类'**) *#执行分类并打印分类结果* **else**:  
 print(testEntry,**'属于非侮辱类'**) *#执行分类并打印分类结果* testEntry = [**'stupid'**, **'garbage'**] *#测试样本2* thisDoc = np.array(setOfWords2Vec(myVocabList, testEntry)) *#测试样本向量化* **if** classifyNB(thisDoc,p0V,p1V,pAb):  
 print(testEntry,**'属于侮辱类'**) *#执行分类并打印分类结果* **else**:  
 print(testEntry,**'属于非侮辱类'**) *#执行分类并打印分类结果***if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 testingNB()

**实验结果：**

