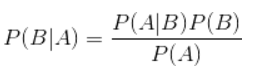
**贝叶斯理论**

贝叶斯理论指的是，根据一个已发生事件的概率，计算另一个事件的发生概率。贝叶斯理论从数学上的表示如下：



,在这里A和B都是事件， P(B)不为0。

* 基本上，只要我们给出了事件B为真，那么就能算出事件A发生的概率，事件B也被称为证据。
* P(A)是事件A的**先验**（先验概率，例如，在证据之前发生的概率）。证据是一个未知事件的一个属性值（在这里就是事件B）。
* P(A|B)是B的后验概率，例如在证据之后发生的概率。

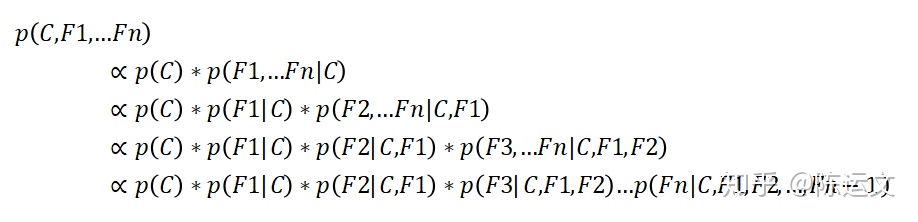
**朴素假设**

现在是时候为贝叶斯理论添加假设了，也就是每个特征之间都是相互独立的。所以我们可以将证据分成每个独立的部分。

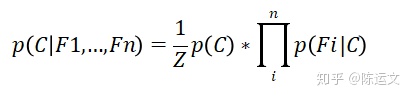
如何两个事件A和B是相互独立的，那么有：

P(AB)=P(A)P(B)

有了特征相互独立的条件以后，对于,联合分布模型可表达为：



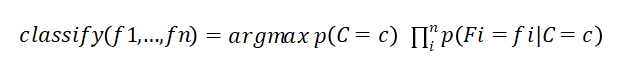
这就意味着，变量C的条件分布可以表达为：



其中，Z只依赖  ,当特征变量已知时Z是个常数。

至此，我们我们可以从概率模型中构造分类器，朴素贝叶斯分类器包括了这种模型和相应的决策规则。一个普通的规则就是选出最有可能的那个：这就是大家熟知的最大后验概率（MAP）决策准则。

相应的分类器便是如下定义的公式：



当特征值为离散型时，类的先验概率可以通过训练集的各类样本的出现次数来估计；当特征值为连续型时，通常的假设这些连续数值为高斯分布。

**对数据集的分析**

数据集为54个维度的数据，都是按01分类，这里用最后的is\_spam属性进行分类。

package com.example.test;

import weka.classifiers.Evaluation;

import weka.classifiers.bayes.NaiveBayes;

import weka.classifiers.evaluation.Prediction;

import weka.core.Instance;

import weka.core.Instances;

import weka.core.converters.ConverterUtils;

import java.util.List;

/\*\*

\* @title: Bayes

\* @Author: Stanton JY

\* @Date: 2022/10/11 14:31

\*/

public class Bayes {

public static void main(String[] args) throws Exception {

ConverterUtils.DataSource source = new ConverterUtils.DataSource("src/main/java/com/example/test/spambase.train.arff");

Instances train = source.getDataSet();

train.setClassIndex(train.numAttributes() - 1); // target attribute: (Sweet)

//build model

NaiveBayes model = new NaiveBayes();

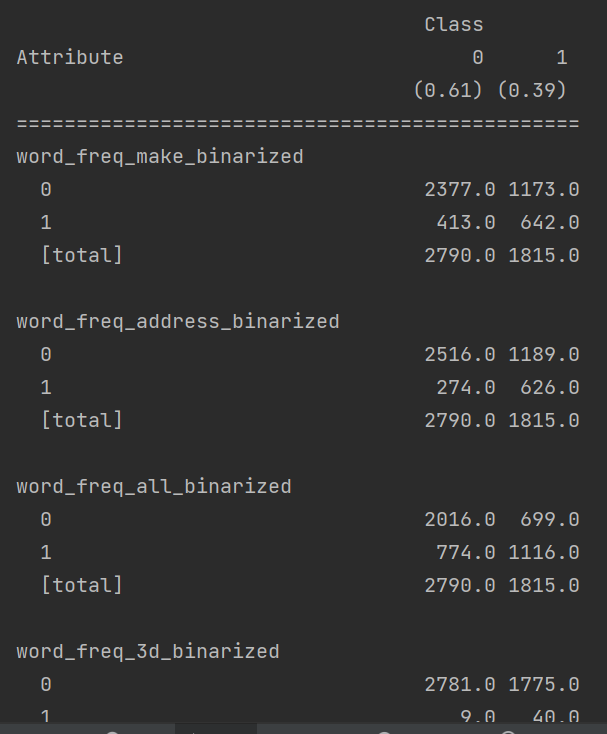
model.buildClassifier(train);

System.out.println(model);

}

}

**实验结果**



如图，数据集被以is\_spam属性分为两类，其中“0类”占0.61。