内容简介

在完成数据准备后，本节通过朴素贝叶斯回归模型，对公司去向进行分类，回答本案例中的业务问题。

1. 主要内容

- 事件、概率和方差分析等统计学相关理论；
- 贝叶斯定理及朴素贝叶斯分类相关理论；
- 特征选择以及 SelectKBest 代码实现；
- 高斯朴素贝叶斯模型的代码实现；
- 从最基础的网格搜索超参数到使用 GridSearchCV 直接实现网格搜索一系列代码实现过程；
- 结合方差分析对最终模型参数进行分析；
- 探究先验参数对分类的影响。

2. 学习目标

学完本节，能解决以下问题：

- 方差分析指的是什么？在最终模型参数中方差分析体现在哪里？
- 特征选择的优点有哪些？SelectKBest 如何通过代码实现？
- 贝叶斯定理是什么？本案例中最终高斯朴素贝叶斯模型中需要设定的先验概率，如何结合朴素贝叶斯定理进行解释？
- 如何使用流水线实现 cross_val_score 交叉验证以及 GridSearchCV 交叉验证进行网格搜索超参数？

统计学的几个相关概念

之前我们已经介绍过统计学中样本和总体的概念，在随机试验中，样本和总体的概念依然适用。以掷骰子为例，每掷一次骰子，就是在进行一次随机试验，对应一个个体。通过对每次试验结果的统计，可以得到不同骰子点数出现的概率。

1. 随机事件

随机事件是概率论中的一个基本概念。在同一组条件下，对某事物或现象所进行的观察或实验叫作**试验**，把观察或试验的结果叫做**事件**。

例如，随意抛掷一颗色子（一个质地均匀、样式对称的正六面体，六面分别刻有1,2,3,4,5,6,六个数字）都是一次试验。色子落地，出现 1 点、2 点.....6 点，或为奇数点、偶数点、点数大于 3 等都是一个事件，而且这些事件都是在一次试验中可能出现也可能不出现的。与此不同的还有两种事件，即在一次试验中，点数小于 7 这一事件，在每次抛掷后是一定出现的，称为**必然事件**，而点数大于 6 这一事件，在每次抛掷后一定不会出现，称为**不可能事件**。

而**随机事件**（random event）与两者不同，是指：

在同一组条件下，每次试验可能出现也可能不出现的事件，因此随机事件也叫偶然事件。

2. 概率

频率是指在相同条件下进行 n 次随机试验，事件 A 出现 m 次，则比值 m/n 称为事件 A 发生的频率。随着 n 的增大，上述频率围绕某一常数 p 上下摆动，且波动的幅度逐渐减小，趋向于稳定，这个频率的稳定值即为事件 A 的**概率**（probability），记为

$$P(A) = \frac{m}{n} = p$$

在案例一中我们介绍过离散型统计分布和连续型统计分布，它们统称为概率分布，实际

上描述的就是“x取某一特定的值（离散型或者连续型）”这一事件发生的概率。因为 x 的取值有多个，因此描述的结果是概率分布。对于连续变量，则有概率密度函数用于描述这个随机变量的输出值的概率的函数。如正态分布、对数正态分布等分布在理论上都有对应的概率密度函数。

条件概率：是指事件 A 在另外一个事件 B 已经发生条件下的发生概率，表示为 $P(A|B)$。

3. 方差分析

方差分析（Analysis of Variance, ANOVA）：是一种特殊的假设检验，目的是从总体上判断多组数据（通常组数 k≥3）的均值μ之间的差异是否显著。

例如，某公司研制出了一种饮料，共有无色、橘色、蓝色三种颜色，饮料除了颜色以外其他都相同。随机抽取五家超市进行统计，得到了三种饮料在五家超市中的销售量记录。为了检验饮料的颜色对销售量是否有影响，设 μ_1、μ_2、μ_3 分别表示三种颜色饮料的平均销售量，则假设检验中的原假设就是 H_0：$\mu_1 = \mu_2 = \mu_3$，备择假设 H_1：μ_1、μ_2、μ_3 不全相等。检验上述假设所采用的方法就是方差分析。

小知识：之所以叫方差分析，是因为虽然人们感兴趣的是均值，但在判断均值之间是否有差异是需要借助于方差。

方差分析中有一些基本术语

术语	解释	颜色对销售量是否有影响
因素	要检验的对象	颜色
水平	因素的具体体现	颜色的不同取值
观测值	在每个因素水平下得到的样本值	每种颜色饮料的销售量

（1）随机误差、系统误差

随机误差：在因素的同一水平下，样本的各观察值之间的差异。比如，同一颜色在 5 家超市的销售量是不同的。这种差异可能是由于抽样的随机性导致的，被称作随机误差。

系统误差：在因素的不同水平下，各观察值之间的差异。比如，同一家超市，不同颜色饮料销售量的差异也可能是由于颜色本身所造成的，这种误差是由系统性因素导致的，称为系统误差。

（2）组内方差、组间方差

两种误差体现为组内方差和组间方差。

组内方差：因素的同一水平下样本数据的方差，只包含随机误差。例如，无色颜料在 5 家超市销售数量的方差，属于无色组内方差，这种差异是抽样的随机性导致的。

组间方差：因素的不同水平下各样本之间方差，包含随机误差和系统误差。例如，无色、橘色、蓝色三种饮料销售量之间的方差，属于组间方差，这种差异可能同时由于抽样的随机性和颜色本身所造成。

方差分析就是比较随机误差和系统误差，以检验均值是否相等。组间方差与组内方差的比值也称为 F 统计量，F 值越大，表明均值之间的差异越显著。在线性回归模型中，我们曾经提过回归方程的显著性检验-F 检验，其原理也是通过计算 F 统计量，检验自变量 x_i 的权重 β_i 是否都等于 0。

贝叶斯定理

贝叶斯定理是朴素贝叶斯模型的理论基础，它的基本内容如下：

设有 N 种可能的类别标记，记为 $y = \{C_1, C_2, ..., C_N\}$，样本 $\boldsymbol{x}$ 用 d 个属性集的测量值描述，即每个样本用一个 d 维属性向量 $\boldsymbol{x} = \{x_1, x_2, ..., x_d\}$ 表示（x_i 为 $\boldsymbol{x}$ 在第 i 个属性上的取值）。对于分类问题，希望给定 $\boldsymbol{x}$ 的属性描述，找出 $\boldsymbol{x}$ 属于类 C_i 的概率（i=1,2,...,N）。

想要求解的 $P(C_i \mid \boldsymbol{x})$ 称为后验概率，即 $P(C_i \mid \boldsymbol{x})$ 表示在条件 $\boldsymbol{x}$ 下，C_i 的后验概率。

例如，假设我们用性别和收入来描述顾客，而 $\boldsymbol{x}$ 是一位女性且收入 1 万的顾客。y 表示顾客是否会购买化妆品。则 $P(C_i \mid \boldsymbol{x})$ 反映当我们知道顾客的性别和收入时，顾客 $\boldsymbol{x}$ 将购买化妆品的概率。

类似的，$P(\boldsymbol{x} \mid C_i)$ 是条件 C_i 下，$\boldsymbol{x}$ 的后验概率。也就是在已知顾客 $\boldsymbol{x}$ 将购买化妆品，该顾客是女性且收入为 1 万的概率。

而 $P(C_i)$ 称为先验概率。对应例子中，它指的是任意给定顾客将购买化妆品的概率，而不管他们的性别收入或任何其他信息。

贝叶斯定理给出了基于以上所提及的概率来计算后验概率 $P(C_i \mid \boldsymbol{x})$ 的方法：

$$P(C_i \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid Ci)P(Ci)}{P(\boldsymbol{x})}$$

朴素贝叶斯分类

朴素贝叶斯分类法预测 $\boldsymbol{x}$ 属于类 C_i，当且仅当

$$P(C_i \mid \boldsymbol{x}) > P(C_j \mid \boldsymbol{x}) \qquad 1 \le j \le N, j \ne i$$

也就是概率 $P(C_i \mid \boldsymbol{x})$是所有类别中最大的时候，就将 $\boldsymbol{x}$ 分类到类 C_i 。

根据贝叶斯公式 $P(C_i \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid Ci)P(Ci)}{P(\boldsymbol{x})}$，由于 $P(\boldsymbol{x})$对所有类为常数，所以只需要 $P(\boldsymbol{x} \mid C_i)P(C_i)$最大即可。

令 D_{Ci}表示训练集 D 中 C_i类样本组成的集合，则可估计 $P(C_i) = |D_{Ci}| / |D|$。但是对于具有许多属性的数据集，计算 $P(\boldsymbol{x} \mid C_i)$可能会非常复杂，为了降低计算开销，朴素贝叶斯模型采用了一种朴素的属性条件独立性假设：对已知类别，**假设所有属性相互独立**（即属性之间不存在依赖关系），也就是假设每个属性独立地对分类结果发生影响。因此：

$$P(\boldsymbol{x} \mid C_i) = \prod_{k=1}^{d} P(xk|Ci) = P(x_1 \mid C_i)\, P(x_2 \mid C_i) \ldots P(x_d \mid C_i)$$

对离散属性而言，令 $D_{ci,xk}$表示 D_{Ci}中在第 k 个属性上取值为 x_k的样本组成的集合，则条件概率 $P(x_k \mid C_i)$可估计为 $P(x_k \mid C_i) = |D_{ci,xk}| / |D_{Ci}|$.

对连续属性而言，通常假定连续属性服从均值为μ、标准差为σ的正态（高斯）分布。对于第 C_i类样本，这一连续属性的取值则服从均值为$\mu_{ci,k}$、标准差为$\sigma_{ci,k}$的正态分布，则根据正态分布的概率密度函数，有

$$P(x_k \mid C_i) = \frac{1}{\sigma ci,k\sqrt{2\pi}}\, e^{-\frac{(x-\mu ci,k)^2}{2\sigma^2}}$$

其中$\mu_{ci,k}$、$\sigma_{ci,k}$分别表示 C_i类样本属性 x_k的均值和标准差。

朴素贝叶斯模型就是在给定训练集中的属性以及分类结果的情况下，分别根据以上公式计算 $P(\boldsymbol{x} \mid C_i)$和 $P(C_i)$，再根据 $P(C_i \mid \boldsymbol{x})$取得最大值的情况，将样本 $\boldsymbol{x}$ 分到 C_i类。

因此，朴素贝叶斯模型要求属性是服从多项式分布的离散变量，或者是服从正态分布的

连续变量。在实践中，有时朴素贝叶斯模型的表现可能不会很好。这主要是因为现实情况中的属性并不一定符合条件独立性假设，属性之间通常会相互影响。比如，例子中顾客的性别和收入通常是有一定相关性的。

本次案例中，通过调用函数 GaussianNB 来实现朴素贝叶斯模型。

朴素贝叶斯不具备案例一和案例二中模型所述的特征选择的功能，也无法通过正则化得到稀疏化的结果，所以我们采用预先特征选择的方法获得高相关特征子集之后进行分类。特征选择采用最基本的 SelectKBest 类，根据某一计分函数选择得分最高的特征集合，需要给定的参数有保留特征数量 K 和计分函数。K 的选择采用网格搜索方法，计分函数选择要根据输入输出类型确定。

朴素贝叶斯为分类模型，目标变量为定类变量，可选的计分函数包括 f_classif，chi2 和 mutual_info_classif 三种。其中，后两者仅用于输入特征非负的条件，且最后一项计算复杂度相对较高，所以案例中采用 f_classif 函数。这一函数对 $\boldsymbol{x}$ 和 y 进行方差分析，计算 F 统计量，以检验不同类之间该特征的均值是否具有明显差异。如果 F 统计量越大，则说明均值存在明显差异，那么这一特征对样本的分类将具有一定影响，可以选择这一特征用于朴素贝叶斯模型中实现分类。

在本次案例中，SelectKBest 的功能就是在给定 f_classif 这一计分函数的情况下，通过方差分析给出 K 个高相关特征，用于后续的模型建立。

特征选择

特征选择是关于怎样选取好的特征，还没有严格、快捷的规矩可循，需要专业领域知识和特征分析经验。通常特征数量很多，但我们只想选用其中一小部分，有如下几个原因：

- 降低复杂度：随着特征数量的增加，很多数据挖掘算法需要更多的时间和资源。减少特征数量，是提高算法运行速度，减少资源使用的好方法。
- 降低噪音：增加额外特征并不总会提升算法的表现。额外特征可能扰乱算法的正常工作，这些额外特征间的相关性和模式没有实际应用价值（这种情况在小数据集上很常见）。只选择合适的特征助于减少出现没有实际意义的相关性的几率。
- 增加模型可读性：根据成千上万个特征创建的模型来解答一个问题，对计算机来说很容易，但模型对我们自己来说就晦涩无比。因此，使用更少的特征，创建我们自己可以理解的模型，就很有必要。

在很多的情况下，最佳特征的选择会随着特征数量的增加，选择复杂度呈指数增长。一个变通方法是：不要找表现好的子集，而只是去找表现好的单个特征（单变量），只要测量变量和目标类别之间的某种相关性就行。

sklearn 中 SelectKBest 用于选择单变量特征的转换器，只保留 k 个最高分的特征，同时基于 sklearn 中 f_classif 方差分析计算单变量的 F 统计量。

结果分析

首先我们根据网格搜索选择超参数，在总体 26 个特征的情况下，效果最好的特征数 k=20，目标变量的先验概率最好为（0.5,05）。此时最好准确率约为 0.6506。

```
最好准确率: 0.6506
最好模型参数设置: {'gnb__priors': (0.5, 0.5), 'selector__k': 20}
```

根据这一参数设定运行模型后得到如下结果，选择后的 20 个特征为注册资本、挂牌日期、营业收入等。获得内部参数表后得到，在不同的类别中个特征的均值和方差如下图所示。

	保留特征	F-value	0类均值	0类方差	1类均值	1类方差
0	listing_days	142.379595	-0.231831	0.799158	0.044601	1.013813
1	registered_capital	127.552477	0.236911	0.900990	-0.032430	1.020230
2	supplier_rt_1st	32.811476	-0.120788	1.018529	0.019257	1.000619
3	total_stock_equity	128.096654	0.238939	0.906180	-0.031209	1.018351
4	business_income	244.602319	0.321321	0.833235	-0.034416	0.914432
5	asset	351.773412	0.386299	0.872051	-0.054235	0.991564
6	liability	401.287494	0.405257	0.876598	-0.063709	0.974746
7	gross_cash_flow	180.498239	0.305020	1.099359	-0.020490	0.883612
8	net_asset_per_share	85.471159	0.228320	1.444740	-0.020525	1.002616
9	earning_per_share	61.181323	0.206543	2.100877	-0.021728	0.776129
10	asset_liablity_rt	89.758664	0.233765	1.973839	-0.042682	0.902272
11	ave_executive_edu	27.244073	0.120151	1.003579	-0.006898	0.997805
12	holder_rt_1st	19.222941	0.090279	0.994193	-0.016530	1.010542
13	supplier_rt_5all	21.797774	-0.096889	1.013720	0.017292	1.006666
14	gross_profit_rt	40.896063	0.142335	0.914429	-0.010006	1.002521
15	net_profit_rt	39.211655	0.139059	0.965923	-0.011571	0.988699
16	roa	90.495247	0.211372	1.123672	-0.025613	0.972742
17	asset_gr	63.166864	0.168283	1.329328	-0.034580	0.871445
18	income_gr	69.175154	0.210599	1.610378	-0.018638	0.955756
19	holder_rt_10all	144.486861	0.960952	0.004575	0.935528	0.010536

F-value 即为 F 统计量，值越高，标明这一特征在两个公司类别下的均值差异更显著，说明根据这一特征来判断公司的最终去向效果会更好。

有 10 个特征 0 类的方差要高于 1 类方差，例如基本每股收益（earning_per_share）、营业收入增长率（income_gr）等。但是除此以外大部分特征，0 类的均值还是较高于 1 类均值。0 类中的公司包含摘牌和转板两类，转板公司通常是往更好的市场流动，这表明离开新三板的公司不一定代表运营不善，恰恰相反，是运行较好的公司离开了三板市场，这种现

象在各大咨询平台均有说明，投资人也越发能意识到这一现象的存在。从方差来看，也可以支持上述结论。

“好”公司离开新三板的根本原因是市场不完善时，信息较为闭塞不够透明，存在公司与投资人之间的信息不对称问题。“好的公司”和“坏的公司”没有得到有效区分，投资人则会给出相对较低的接近平均水平的价格进行投资，“好的公司”则认为市场无法满足其融资需求而离开市场。这在经济学上称作“劣币驱逐良币”现象。

还有一部分是公司发展看好而转板，反应在特征上就是方差更大。这种现象在各大咨询平台均有说明，投资人也越发能意识到这一现象的存在。

新三板作为一个二级市场，接受的公司本身质量就良莠不齐，在信息披露等方面并没有积极跟进，负责信息披露工作的主办券商并不具有动力完成公司辅导工作。整体来说市场的问题相对较大，一一印证了当下投资人和咨询分析师对于当前市场改变现状的说明。

先验概率对分类的影响

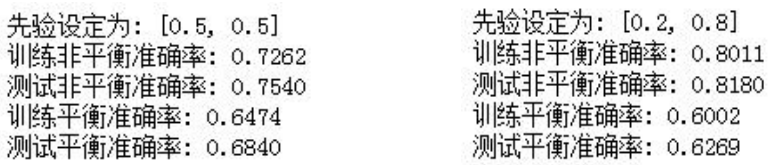
接下来探索一下先验概率的分布对于分类的影响。

我们选择两组先验进行对比：

先验概率：(0.5,0.5)----------平衡分布，表示留存和非留存各占一半；

先验概率：(0.2,0.8)----------非平衡分布，更接近市场实际情况。

下图是拟合模型结果：



①比较非平衡数据集在两种情况下的准确率：

可以看到平衡分布的准确率没有在真实情况下的准确率高；

②比较平衡数据集在两种情况下的准确率：

可以看到平衡分布的准确率比在真实情况下的准确率高。

说明平衡的先验概率分布对于提升平衡准确率来说具有意义，相反真实先验分布有利于分类器分类非平衡的数据集（真实情况下的数据集）。

平衡准确率基本可以看做在每一类中的分类准确率的平均值。这样的差异是合理的，真实（非平衡）先验会使得分类器认为任何一个样例有更大的可能性为 1 类，所以能够提升全局准确度。而平衡先验则表示模型对 0 类和 1 类一视同仁。

不同的实际环境中有不同的要求，因此应该灵活选择先验分布。

特征选择与高斯朴素贝叶斯模型总结

1、方差分析指的是什么？在最终模型参数中方差分析体现在哪里？

（1）在了解方差分析之前，需要建立关于随机事件、概率、随机误差、系统误差等概念基础

在同一组条件下，对某事物或现象所进行的观察或实验叫作试验，把观察或试验的结果叫做事件，事件分为必然事件、不可能事件以及随机事件。

必然事件：在同一组条件下，必然会发生的事件称为必然事件；

不可能事件：在同一组条件下，永远不可能发生的事件称为不可能事件；

随机事件（偶然事件）：在同一组条件下，每次试验可能出现也可能不出现的事件。

概率的概念相对简单：

随着随机试验次数的增大，频率围绕某一常数 p 上下摆动，且波动的幅度逐渐减小，趋向于稳定，这个频率的稳定值即为某事件的概率。

但再次基础上需要了解之前介绍的离散型统计分布和连续型统计分布都统称为概率分布，其中连续型变量并不像离散型变量，可以通过 x 的多个取值体现出来的分布形状，而是有对应概率密度函数来描述连续型统计分布。

同时还需要了解条件概率这一概念，它是贝叶斯定理的基本概念，及事件 A 在另外一个事件 B 已经发生条件下的发生概率，表示为 $P(A|B)$。

随机误差和系统误差、组内方差和组间方差：

随机误差：在因素的同一水平下，样本的各观察值之间的差异。这种差异可能是由于抽样的随机性导致的。

系统误差：在因素的不同水平下，各观察值之间的差异。这种误差是由系统性因素导致

的。

组内方差：因素的同一水平下样本数据的方差，只包含随机误差。

组间方差：因素的不同水平下各样本之间方差，包含随机误差和系统误差。

（2）方差分析

方差分析就是比较随机误差和系统误差，以检验均值是否相等。具体是通过计算组间方差与组内方差的比值，组间方差与组内方差的比值也称为 F 统计量，F 值越大，表明均值之间的差异越显著。在线性回归模型中，我们曾经提过回归方程的显著性检验-F 检验，其原理也是通过计算 F 统计量，检验自变量 x_i 的权重 β_i 是否都等于 0。

（3）方差分析在最终模型参数中的体现

通过一系列网格搜索超参数得到 20 个特征模型效果最佳，而 20 个特征的 F-value 即为 F 统计量，F 值越大，表明均值之间的差异越显著，我们可以对这些统计量进行排序，可以发现 F 统计量前三正是 liability、asset、business_income 三个字段。其次 0 类方差和 1 类方差分别对应取值“0”时的组内方差和取值“1”时的组内方差。

方差分析的目的在于检验均值是否相等，我们发现大部分特征，0 类的均值还是较高于 1 类均值。0 类中的公司包含摘牌和转板两类，转板公司通常是往更好的市场流动，这表明离开新三板的公司不一定代表运营不善，恰恰相反，是运行较好的公司离开了三板市场，这种现象在各大咨询平台均有说明，投资人也越发能意识到这一现象的存在。

2、特征选择的优点有哪些？SelectKBest 如何通过代码实现？

（1）特征选择的优点

- 降低复杂度：随着特征数量的增加，很多数据挖掘算法需要更多的时间和资源。
- 降低噪音：增加额外特征并不总会提升算法的表现。

❖ 增加模型可读性：使用更少的特征，创建我们自己可以理解的模型，就很有必要。

在很多的情况下，最佳特征的选择会随着特征数量的增加，选择复杂度呈指数增长。一个变通方法是：不要找表现好的子集，而只是去找表现好的单个特征（单变量），只要测量变量和目标类别之间的某种相关性就行。

（2）SelectKBest 的代码实现

SelectKBest()是 sklearn.feature_selection 库中的一个函数，设置参数有“score_func”和“k”，第一个参数为计分函数，包括：f_classif，chi2 和 mutual_info_classif，两者仅用于输入特征非负的条件，且最后一项计算复杂度相对较高，所以案例中采用 f_classif 函数。k 值为人为设定取用字段个数，代码如下图所示：

```
# 导入特征选择工具SelectKBest和计分函数f_classif
from sklearn.feature_selection import SelectKBest, f_classif

# 设置特征选择参数
selector = SelectKBest(score_func=f_classif, k=3)
# 调用fit_transform，完成平衡训练集X_train_bal的训练和转换
X_train_bal_new = selector.fit_transform(X_train_bal, y_train_bal)
# 调用transform，对测试集进行转换
X_test_new = selector.transform(X_test)
```

除此以外，我们还可以实现以下操作：

①查看哪些特征被保留下来

```
# 查看哪些特征被保留下来
selector.get_support()
```

```
array([False, False, False, False, False, False,  True,  True,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False])
```

②使用布尔型索引找到保留的特征

```
# 所有特征名称
names = np.array(x_mapper.transformed_names_)
# 设置布尔型索引
mask = selector.get_support()
# 使用布尔型索引找到保留的特征
names[mask]
```

```
array(['business_income', 'asset', 'liability'], dtype='<U19')
```

③使用布尔型索引找到保留特征的 F 统计量

```
# 使用布尔型索引找到保留特征的F统计量
selector.scores_[mask]
```

```
array([244.60231857, 351.77341199, 401.28749365])
```

④所有特征的 F 统计量

```
# 所有特征的F统计量
selector.scores_
```

```
array([1.42379595e+02, 1.27552477e+02, 6.93801514e+00, 1.56855494e+00,
       3.28114765e+01, 1.28096654e+02, 2.44602319e+02, 3.51773412e+02,
       4.01287494e+02, 1.80498239e+02, 8.54711585e+01, 6.11813230e+01,
       8.97586644e+01, 3.73016902e+00, 1.49197227e+01, 2.72440731e+01,
       1.92229407e+01, 2.63593846e-01, 2.17977739e+01, 4.08960631e+01,
       3.92116548e+01, 9.04952466e+01, 6.31668635e+01, 6.91751540e+01,
       1.09028873e+01, 1.44486861e+02])
```

3、贝叶斯定理是什么？本案例中最终高斯朴素贝叶斯模型中需要设定的先验概率，如何结合朴素贝叶斯定理进行解释？

（1）贝叶斯定理

设有 N 种可能的类别标记，记为 $y = \{C_1, C_2, ..., C_N\}$，样本 $\boldsymbol{x}$ 用 d 个属性集的测量值描述，即每个样本用一个 d 维属性向量 $\boldsymbol{x} = \{x_1, x_2, ..., x_d\}$ 表示（x_i 为 $\boldsymbol{x}$ 在第 i 个属性上的取值）。对于分类问题，希望给定 $\boldsymbol{x}$ 的属性描述，找出 $\boldsymbol{x}$ 属于类 C_i 的概率（i=1,2,...,N）。

贝叶斯定理主要基于该公式：

$$P(C_i \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid Ci)P(Ci)}{P(\boldsymbol{x})}$$

$P(C_i \mid \boldsymbol{x})$ 称为：在 x 条件下 C_i 的后验概率。

该条件概率建立：已知条件 C_i 下，$\boldsymbol{x}$ 的后验概率 $P(\boldsymbol{x} \mid C_i)$，以及先验概率 $P(C_i)$ 和 $P(\boldsymbol{x})$。

在给定的训练集中先验概率等于类标签取值的频率，例如案例未重构训练集前中：

0 类标签：1 类标签=537:3379，则

先验概率 $P(C_0)$=537/（537+3379）≈0.1371；$P(C_1)$=3379/（537+3379）≈0.8629

数据重构后，训练集的先验概率 $P(C_0)=P(C_1)$=0.5。

当然这个先验概率可以人为既定，默认取值为上述计算过程。

结合朴素贝叶斯分类文档中的介绍，对于离散型变量，给定 x 取值，频率即为概率；对于连续型变量，给定 x 取值，通过服从于均值为$\mu_{ci,k}$、标准差为$\sigma_{ci,k}$的正态分布概率密度函数进行求解即可得到。

（2）先验概率结果分析

为探索先验概率的分布对于分类的影响，除了默认的（0.5,0.5）外，根据咨询专家得到真实情况下的先验概率（0.2,0.8），分别求解最终模型在两种情况下对于平衡数据集和非平衡数据集的准确度。

可以发现：非平衡数据集中，平衡分布的准确率没有在真实情况下的准确率高，反之平衡数据集中，平衡分布的准确率比在真实情况下的准确率高，进一步说明真实（非平衡）先验会使得分类器认为任何一个样例有更大的可能性为 1 类，所以能够提升全局准确度。而平衡先验则表示模型对 0 类和 1 类一视同仁。

但是不同的实际环境中有不同的要求，应该灵活选择先验分布。

4、如何使用流水线实现 cross_val_score 交叉验证以及 GridSearchCV 交叉验证进行网格搜索超参数？

（1）首先需要了解本案例**高斯朴素贝叶斯模型**构建的代码实现过程，如下图所示：

```
# 导入GaussianNB
from sklearn.naive_bayes import GaussianNB
# 创建高斯朴素贝叶斯模型
clf = GaussianNB(priors=[0.5, 0.5])

# 用特征选择后的平衡训练集训练模型
clf.fit(X_train_bal_new, y_train_bal)

# 用特征选择后的测试集测试模型
y_pred = clf.predict(X_test_new)
```

模型评估使用分类的准确度：平衡准确度

```
# 导入平衡准确度函数
from sklearn.metrics import balanced_accuracy_score

# 评估模型在特征选择后的测试集上的表现：计算平衡准确度
balanced_accuracy_score(y_test, y_pred)
```

```
0.6498079107804112
```

高斯朴素贝叶斯模型只需设置先验概率这一个参数，其次通过 balanced_accuracy_score 函数计算模型的平衡准确度。

（2）其次**结合交叉验证搜索超参数：**

超参数组合中包含：先验概率、特征选择中 k 值

首选选用之前学过的 cross_val_score()函数通过设置 cv 参数，直接进行数据分割并得到每组中的分值然后求均值。代码实现如下图所示：

```
from sklearn.model_selection import cross_val_score
# 新建DataFrame保存搜索结果
result_frame = pd.DataFrame(columns=['k', 'priors', 'score'])

# 穷举k和priors的组合
for k in [5, 10, 15, 20]:
    for priors in [(0.5, 0.5), (0.2, 0.8)]:
        # 设置特征选择参数
        selector = SelectKBest(score_func=f_classif, k=k)
        # 调用fit_transform，完成平衡训练集X_train_bal的训练和转换
        X_train_bal_new = selector.fit_transform(X_train_bal, y_train_bal)
        # 调用transform，对测试集进行转换
        X_test_new = selector.transform(X_test)

        # 创建高斯朴素贝叶斯模型
        clf = GaussianNB(priors=priors)
        # 使用交叉验证
        scores = cross_val_score(clf, X_train_bal_new, y_train_bal, scoring='balanced_accuracy', cv=3)
        score = np.mean(scores)
        # 将参数搜索结果放入结果表中
        result_frame = result_frame.append({'k':k, 'priors':priors, 'score':score}, ignore_index=True)

print("以下为不同参数设定下不同指标的运行结果：")
print(result_frame)
```

```
以下为不同参数设定下不同指标的运行结果：
    k      priors     score
0   5  (0.5, 0.5)  0.638206
1   5  (0.2, 0.8)  0.586118
2  10  (0.5, 0.5)  0.648267
3  10  (0.2, 0.8)  0.611720
4  15  (0.5, 0.5)  0.644863
5  15  (0.2, 0.8)  0.593223
6  20  (0.5, 0.5)  0.650636
7  20  (0.2, 0.8)  0.597071
```

案例中暂且设置为 3 折交叉验证并不做拓展，有兴趣的同学可以试试其他的 cv 值。cross_val_score 函数中参数“score”还有设置哪些计分函数？可以参照该官方文档：https://scikit-learn.org/stable/modules/model_evaluation.html

其次 GridSearchCV 也是交叉验证的另一种方法，它的优势在于非常友好于网格搜索超参数及流水线的创建，可以直接给出最优超参数组合，更加简便。使用方法与 cross_val_score()函数基本一致，不同在于需要将模型遍历的参数定义为字典，然后存放在 GridSearchCV 括号中。例如案例中代码：

```
from sklearn.model_selection import GridSearchCV
# 新建DataFrame保存搜索结果
result_frame = pd.DataFrame(columns=['k', 'priors', 'score'])

# 穷举k的组合
for k in [5, 10, 15, 20]:
    # 设置特征选择参数
    selector = SelectKBest(score_func=f_classif, k=k)
    # 调用fit_transform，完成平衡训练集X_train_bal的训练和转换
    X_train_bal_new = selector.fit_transform(X_train_bal, y_train_bal)
    # 调用transform，对测试集进行转换
    X_test_new = selector.transform(X_test)

    # 将需要遍历的参数定义为字典
    params = {'priors':[(0.5, 0.5), (0.2, 0.8)]}
    # 创建高斯朴素贝叶斯模型
    clf = GaussianNB(priors=priors)
    # 定义网格搜索中使用的模型和参数
    grid_search = GridSearchCV(clf, params, scoring='balanced_accuracy', cv=3)
    # 使用网格搜索模型拟合数据
    grid_search.fit(X_train_bal_new, y_train_bal)
    # 交叉验证最高分
    score = grid_search.best_score_
    # 模型最优参数
    priors = grid_search.best_params_

    # 将参数搜索结果放入结果表中
    result_frame = result_frame.append({'k':k, 'priors':priors, 'score':score}, ignore_index=True)

print("以下为不同参数设定下不同指标的运行结果：")
print(result_frame)
```

```
以下为不同参数设定下不同指标的运行结果：
    k                  priors     score
0   5  {'priors': (0.5, 0.5)}  0.638207
1  10  {'priors': (0.5, 0.5)}  0.648269
2  15  {'priors': (0.5, 0.5)}  0.644865
3  20  {'priors': (0.5, 0.5)}  0.650636
```

其中 params 为定义字典，是 GridSearchCV 中参数 param_grid 的取值，通过.best_score_和.best_params_得到交叉验证最高分和模型的最优参数。

注：一般情况(param_grid = params)中(param_grid =)可以省略，在模型后直接写定义字典即可。

（3）使用流水线进行网格搜索

在了解了流水线和网格搜索的创建后，将两者结合起来便可以实现使用流水线进行网格搜索，例如案例中的代码：

```
from sklearn.model_selection import GridSearchCV
# 创建流水线
pipe = Pipeline([('selector', SelectKBest(score_func=f_classif)),
                 ('gnb', GaussianNB())])
# 设置参数字典
param_dict = {'selector__k':[5, 10, 15, 20],
              'gnb__priors':[(0.5, 0.5), (0.2, 0.8)]}
# 将流水线加入网格搜索
grid_search = GridSearchCV(pipe, param_grid=param_dict, scoring='balanced_accuracy', cv=3)
# 对训练集进行拟合
grid_search.fit(X_train_bal, y_train_bal)
# 交叉验证最高分
best_score = grid_search.best_score_
# 模型最优参数
best_param = grid_search.best_params_

print("最好准确率: %.4f" % best_score)
print("最好模型参数设置:", best_param)
```

```
最好准确率: 0.6506
最好模型参数设置: {'gnb__priors': (0.5, 0.5), 'selector__k': 20}
```

需要注意的是，创建流水线需要使用 Pipeline 方法，因为流水线中的步骤需要从字典中获取取值，如果使用 make_pipeline，就无法根据步骤名称获取参数取值。并且定义字典时 k 值和 priors 前需要两个下划线，意指定位到名称后的步骤，即步骤中参数的取值。其次将定义好的字典赋值给 GridSearchCV 中参数“param_grid=”。这样就实现 GridSearchCV 方法的流水线进行网格搜索。

补充：

sklearn 中经常会出现诸如：

```
# 所有特征的F统计量
selector.scores_
```

```
# 所有特征名称
names = np.array(x_mapper.transformed_names_)
```

```
# 交叉验证最高分
score = grid_search.best_score_
# 模型最优参数
priors = grid_search.best_params_
```

等属性都是以下划线为结尾的，这是 sciki-learn 的一个特点，它总是用下划线作为来自训练数据集的属性的结尾，以便将他们与由用户设置的参数区分开。