

订阅DeepL Pro以编辑此演示文稿。  
访问[www.DeepL.com/pro](https://www.deepl.com/pro?cta=edit-document)，了解更多信息。

没关系我们。我们好吗？是的 你们有充分的时间讨论 Yes.你们有充分的时间讨论好的我记得上周我从这里开始，我说我这周从这边开始。我还记得在这边你有什么建议？优点缺点有什么有趣的发现吗？很容易解释当然我是说，这和线性回归很相似。我们有可解释性。这是一个大问题。这也是逻辑回归在信用评分中如此流行的主要原因之一。因此，在信用评分中，出于监管原因，你必须能够解释一个模型是如何决定一个人是否值得信贷的。因此，你不能只使用黑箱模型。人们通常认为，只要把信用评分数据扔进神经网络，就会有好东西出来。但在银行里你不能这么做，因为如果监管机构打电话来告诉你，告诉你，告诉我你是怎么计算的，而你基本上是给他们一段代码，然后设置模型告诉我这么做，那就不够好了。你不能这样做决定。所以，是的，可解释性易于解释。非常重要的一点。我们前面有什么东西吗？也许像使用逻辑回归的概率那样。通常它是用来分析历史数据以预测未来的。所以我认为只有使用逻辑回归才有可能。预测概率。所以，是的，这与可解释性有关，不是吗？所以，有了概率这个概念，我们就可以决定一个阈值。例如，我们决定概率阈值应为 0.5，但我们仍会计算不同客户违约的概率。然后，我们使用 sigmoid 函数将其转换为 0 或 1 的结果。所以是的，从解释学的角度来看，概率概念是有道理的，不是吗？你明白我的意思吗？我打了个问号，你反而会记住我的话。我居然不擅长把我的话写成文字。我知道一个学者应该这样 但我不擅长好吧后面有什么东西吗？我想，它的一个缺点是需要专家知识来评估结果，因为你必须选择什么是最好的阈值。你要么想把整体错误率降到最低，要么想把个体错误率降到最低。是啊。所以专家知识，很明显你的很多模型都需要。但我们确实讨论过这个问题。选择阈值和决定哪种误差更严重取决于应用环境。因此，我们在讨论这种银行成本保险时，你必须能够理解银行是如何发生成本的。因此，作为一个外部机构，向他们介绍一些东西会比较困难。所以，是的，非常好。最后一个人，我明白你的意思。正如我们所讨论的，这既是优势也是劣势。因为我们没有看到那么多。呃，你的意思是在结果方面的异常值？是的，非常多。所以是二元结果所以，是的，你有更少的那种 更少的变化或更少。因变量的变化更少是的，完全正确。我的意思是，很明显，你的解释变量中仍有异常值，但这确实提高了模型实际实施的难度。完全正确是的是的我的意思是，我不确定这是否是一个缺点，但如果我们不想，二元结果就不是真正的模型，对吗？是的，它有很大的局限性。所以我提到了我们有多类逻辑回归，我们有所有的逻辑回归特例。但如果我们只谈今天讨论的模型，我们只有二元结果。因此变化较少。因此，既有好的结果，也有坏的结果，变化较少。因此，我们有两个类别，这很容易解释模型，但也更受限制。因此，举例来说，当我们讨论概率概念时，我们报告的并不是真正的违约概率。早些时候，我们报告的是二元结果。因此，我们不知道某人是否违约，例如，是否只是轻微违约，下次就不会了。我们只知道他们违约了。我们对此无能为力。因此，从这个意义上说，这有点限制。中间的东西？是的我们？我想你还提到了这个。优点是速度慢。所以这是一个相对便宜的模型。这也是原因之一。与线性回归类似，它通常是我们在数据上训练和测试的第一个模型，因为它给了我们一个很好的阈值。因此，如果你有一个分类问题，如果你有一个二元分类问题，你可能会尝试实现的第一个模型是逻辑回归，然后你可以使用更高级的模型，这取决于你的数据集有多大。是的，我还有一个与之密切相关的模型，但我要看看是否有人真的告诉了我，而不需要我告诉你。所以，我们在中间，也许在后面还有别的东西。一切都已经安排好了。所以我说变量是。我们假设是也不是。这可能是一个缺点。我特别提到了线性关系你还记得是什么吗？记得所以线性关系是指你的输出与因变量的对数概率之间的关系。因此，因变量值和结果变量之间并不是直接的线性关系。而是概率、变量几率和概率之间的关系。这种情况是否经常发生？因变量值和因变量的对数是否是一种常见的情况？我认为这是我们经常做的一个假设，至少在某种意义上是这样。因此，它基本上告诉我们的是，例如，违约的概率。所以我说违约概率是线性的，线性关系与解释变量有线性关系。所以这是一种。是的。事件的对数概率。事件。对因变量。所以因此，如果我们说概率与变量组合呈线性关系，这在很多情况下都是可以假设的。是的，效果出奇的好。所以，这显然取决于你的应用环境。但以信用评分为例，逻辑回归是一种非常非常非常流行的方法。因此，你或许可以认为，这就是实际中正在使用的方法，因为它的计算成本很低，而且效果出奇的好。前面有什么问题吗？我觉得我们可以补充的是，这真的很简单。我们实际上需要为自己设置价格的参数。是啊，我想这也是它被用于其他神经网络的原因之一。是啊例如，你需要关心和担心的调整参数相对较少。所以有些模型要复杂得多。如果你考虑神经网络，同样，与逻辑回归相比，它们的调整过程极其复杂，但却很容易实现，也很容易解释。边上有什么东西吗？是的。因此，与线性回归相比，我们可以使用分类变量进行预测，因为我们使用的是二进制输出，即 0-1。但在讨论的过程中，我们也意识到，与线性回归相比，我们可能会经常面临多重共线性的问题。因为如果我们预测的是分类变量，我们就会有很多假编码，然后我们也必须管理这些假编码。因此，管理分类变量可能会带来不利。由于自动编码和多重共线性，你必须更加小心，但我们可以预测分类变量。因此，预测分类变量，例如，在这种情况下，二元分类是的，预测分类结果。是的，你必须检查多重共线性。老实说，这是个很糟糕的词。所以是的。是的，它基本上意味着某些变量是其他变量的线性组合，所以它们之间是相互关联的。我们在讨论虚拟变量时，我想这就是一个很好的例子。因此，如果你有三种颜色选项，例如蓝色、红色和黄色，那么如果你试图为所有这三种颜色创建一个虚拟变量，你的模型就会因为多重共线性而失效。原因在于，你完全可以通过某人选择红色、红色还是蓝色的信息来解释他是否选择黄色。因此，你可以通过其他变量的组合来完美解释其中一个变量。这样说有意义吗？这是否意味着这些变量不能必须相互区别。是的。相乘相等没错没错所以它们必须相互独立。所以，比如说，你不能把收入作为一个变量，然后把收入除以2作为另一个变量。我不知道你为什么要这么做，但你可能会这么做。所以，让我们做一个更好的例子。比方说，我们有一个变量叫 "家庭规模"，你的家庭中有多少人，你有一个变量叫 "收入"。你的家庭有多少收入？你不可能有一个变量来告诉你每个家庭成员的收入是多少。因此，举例来说，你不能有一个变量来描述每个家庭成员的价值，因为你完全可以通过将收入除以家庭成员来计算。因此，它们必须相互独立。它们不能相互计算。是的所以这很完美。多重共线性可以有我是说小数这是一个完美线性共线性的例子。但也有可能存在不完美的共线性，这仍然会给数据带来问题。所以你必须检查预测变量的独立性。是的，非常重要。线性回归和逻辑回归都是如此。尽管如此，逻辑回归在假设方面更加灵活。例如，我们对线性回归的一些假设是，误差项必须呈正态分布，并且在所有样本中具有相同的方差。这就是所谓的同方差。我知道统计学家强加给我的另一个可怕的词，那就是误差项的同方差性及其正态分布。逻辑回归不需要满足这两点，但线性回归必须满足这两点。因此，逻辑回归在这方面更灵活一些。但有些假设（如多重共线性）对两者都适用。还有什么要补充的吗？知道你的同事们所涉及的一切。我想说可能没有。是的。这取决于你到底想在时间序列中预测什么。举例来说，如果你只想预测时间序列的方向，这可能是可行的。所以我在想一个非常简单的模型，你想预测一只股票的方向。例如，股票价格。它是上涨还是下跌将是一个二元预测。这有多大价值，我不知道。所以，对于时间序列来说，它可能用处不大。你可以说是或不是，然后将其汇总为一个数字，再将其用作时间序列，也许就像这样。就是这样所以我觉得也许有办法，但这不太适合时间序列分析。我同意是的，但你可以。所以可能适用性有限，也许取决于你的情况，因为结果是二元的，而且我们与对数几率之间是线性关系。还有一件事还有一件事。我们为什么要用对数赔率？哇，这和我们想要进行逻辑转换有关。这就是为什么我们要使用 sigmoid 函数转换概率，以适应这种二元结果。因此，我们采用对数变换使其符合我们的曲线。这就是非常简单的解释，因为我们记得这是对数函数，我们将其计算为指数，而指数取自我们的对数值，因为它们是有意义的。因此，如果我们想将 P 值转化为对数函数来绘制曲线或进行描述，这样做有意义吗？你可以说还没有。我再解释一遍。还没有。很好 谢谢 Very good.谢谢我们的目标是让我们从后面总结一下。如果这是我们的目标，我们希望用它来描述我们的结果，因为我们感兴趣的是某件事情的概率是零还是一。因此，我们要将其转化为二元结果。我们说这个 sigmoid 函数、这个 logistic 函数非常适合。比方说，我们希望用 logistic 函数来建立数据模型，而 logistic 函数看起来就像顶部的这个。因此，我们希望能够用 logistic 函数来预测、模拟 Y 轴上的概率。现在，我们如何通过这些步骤得出这个 Logistic 函数。这就是我们的对数函数。因此，如果我们在计算中后退一步，我们就想把它除掉。这就是我们的线性组合。让我在屏幕上演示一下。在另一边，你也可以看到。这是截距和解释变量的线性组合。你可以看到这里有指数值。这就是指数值，对吗？因为我有德语和英语，用两种语言来表达非常困难。所以，如果我们想在不使用指数的情况下描述这个问题。你就必须从等式的另一面取锁。这就是为什么我们要锁定几率。这样说更有意义吗？很好 很高兴 Good.很好是的，是的。我们可以休息一下了不好意思每天回来五分钟吧可以吗？好的 谢谢 Good.谢谢 - 好的 - Thank you.谢谢 - 好的好吧，每个人都安顿下来。请坐我们将继续正则化我还想谈谈模型调整、数据分割和重采样。如果我们今天不讨论不平衡问题，我将把它移到下一讲。所以不用担心，我们会涵盖所有材料。不过，可能只是因为我说得太多或类似的原因，我才会挪动一下。在我们开始之前，我还想在优缺点部分补充一点。我们的计算成本很低。逻辑回归的优势之一在于，它能很好地处理相对较小的样本。因此，与更复杂的模型相比，逻辑回归或线性回归并不需要庞大的样本。这是因为我们假设的关系相对简单，所以我们不需要大量数据来解释简单的关系。因此，如果样本量较小，有些模型就会比其他模型更好用。逻辑线性回归（logistic linear regression）就是其中一种对小样本非常有效的模型。然后我们还有像随机森林、神经网络和 svms 这样的模型，它们在样本量较大的情况下效果更好。所以。根据你的数据情况，显然会影响你要选择的模型。所以我想把这一点作为另一个优势补充进来。好的。我们来谈谈正则化。当我们决定如何实际拟合模型时，我们讨论了一些事情。在拟合模型时，我们关心什么？在这种情况下，当我说拟合时，我指的是估计我们的更好值，例如逻辑回归中的更好值。很明显，其中一个因素就是预测准确性。所以这里说的是一个非常重要的部分。如果 N 不比 P 大很多，这里我说的 N 是样本或观测值的数量，P 是预测因子的数量。因此，从根本上说，你有多少个 X 值，就会有多少变数。方差显然是合适的。因此，如果我们的样本量与预测因子的数量相比特别小，那么这可能会导致过度拟合，并在以后的建模中出现预测不佳的情况。另一方面，如果预测因子的数量实际上大于观测数据的数量，那么最小二乘法拟合方法的方差就是无限大的。因此，你根本无法使用最小二乘法。如果你有更多的预测因子，比如 X 中有更多的列，那么你就会有样本，这实际上是一个相当大的问题。我们稍后会看到在这种情况下可以做些什么。其中一些方法包括，例如，选择你想要包含的变量，因此大部分情况下都会强制减少预测因子的数量。因此，我们通常可以尝试通过对模型进行正则化等方法来减少方差。还有一点是，在很多情况下，我们都很在意模型的可解释性。这也是我们在线性回归和逻辑回归中都提到过的一点。在很多情况下，能够解释你的结果真的非常非常有价值。因此，如果我们的模型中有很多变量、很多预测因子，就会让我们不必要地拥有一个过于复杂的模型，从而更难解释这种关系中究竟发生了什么。因此，在这种情况下，我们还是要删除其中的几个变量，以便更有效地解释我们的模型。因此，这种特征选择或变量选择的基本思想是，我们要删除无关变量，或者至少是那些对我们的回归模型没有太多解释力的变量。而实现这一目的的一种方法就是所谓的套索回归（lasso regression）。套索回归实际上是一种相对简单的方法，我们可以说，我们希望对模型中预测因子的数量进行惩罚。因此，它与惩罚 R 方中预测因子数量的想法类似。你还记得我们讨论过 R 方值和调整 R 方。在优化模型时，我们也要对预测因子的数量进行惩罚。所以，你应该还记得最小二乘估计的第一部分。因此，如果你有一个多元线性回归，你会记得我们想要最小化误差，误差可以计算为实际值减去。然后这里的第二部分基本上就是你的估算。这就是你的贝塔帽。这个项就是你的误差。也就是你的误差平方和。这就是我们在最小二乘法中要优化的结果。现在我们在最后加上这个红色项。它的作用是计算预测因子的数量，或者说计算你希望模型中包含的预测因子的强度或数量。因此，我想让大家重点关注的是这个 lambda。这个 lambda 表示惩罚项的强度。因此，lambda 越高，每增加一个预测因子，就会增加整个项的强度。如果我们想最小化这个数量来优化我们的模型，你可以看到，每增加一个 lambda 都会使这个项的最小化效果变差。因此，如果我们试图将其最小化，那么每增加一个对不起，每增加一个更好的值，都会使整个项变得更糟。所以你可以根据你的专业知识来选择这个 lambda 值，我知道你希望你的模型有多复杂。我们之前说过，如果你的模型中预测因子多于观测值，那么这个模型就会失效。因此，在这种情况下，你会希望预测因子的数量至少低于观测值。这就是选择 lambda 的一种方法。就是这样。lasso 回归的目的实际上是，我们希望建立一个完美或更好的多元回归模型，其中只包含最重要的预测因子。这就是套索回归的目标。因此，它仍然是一种最小平方和方法，但我们仍然要最小化，但我们现在还要最小化预测因子的数量。因此，如果 lambda 足够大，这实际上会迫使我们的一些系数估计值为零。因此，如果我们的 lambda 足够大，那么我们在这里求和的一些预测因子就必须为零，这实际上就从你的记录中删除了一个变量。所以，这就是你的 x 和贝塔数。你会记住它们的所有系数。因此，如果你想找到一组完美的系数，同时考虑到每一个系数都会使我们的结果变得更糟，那么很明显，其中一些系数的最佳值为零，如果你的最佳值为零，那么这个变量就会从模型中删除。因此，选择一个好的 lambda 值是很重要的。我们通常会使用某种交叉验证方法，因此我们会检查多个 lambda，然后根据某种误差测量、准确度测量等指标，基本上选择出最佳模型。因此，我们最终会根据模型性能来评估不同的 Lambda 值。但你也要记住，例如，预测因子的数量必须要少，以保证可解释性或低于观测数据的数量。是的。这将消除几个变量。没错。没错所以，实际上你也可以同时剔除多个变量。这取决于他们的更好值有多大。因此，即使更好值仍然很高，也不会从模型中移除，因为它仍然是一个非常有影响力的变量，但会被压缩。所以，我总是把它们想象得很大，稍微压缩一下，其中一些变量因为仍然非常重要，所以会很坚挺，而另一些变量则被压缩得很厉害，以至于消失了。因此，我总是把 Lasso 想象成一种压缩方法。我觉得这个名字也可以看作是一种套索，你用绳子套住最重要的变量。重要。变量，如果你想这么理解的话。但没错，它可以从中删除多个变量。所以，是的，它的另一种表述方式就是这个词。所以我们还是在最小化你的误差因此，我们仍在最小化平方误差，即通过调整我们的击球手，使我们的击球手总和小于我们设定的某种阈值。因此，这是一个类似的公式，只是用一个 S 参数代替了 lambda，但其结果是一样的。结果是的，结果是一样的。这只是一个优化最小化公式。是啊很明显你做得更好数值可能取决于不同预测因子的比例。所以，一定要在预测因子标准化后再应用套索回归。这是你做任何优化时的经验之谈。因此，通常情况下，首先要对所有数据进行标准化，因为这有助于模型的实际拟合，并找到最佳的选择。除此以外，还有一个替代方案，那就是富回归。因此，这实际上是将所有系数向零惩罚，但不会将任何系数设置为零。因此，这是一种比较柔和的方法。它仍然会压缩所有数据，但没有任何数值会真正消失。因此，这种方法并不适合用于变量选择或特征选择，但仍可用于解释模型，因为它仍会压缩其中的一些值。然后弹性网是两者的结合。嗯。是啊所以我认为套索回归基本上是一种更突出的方法。所以这也是我们明天要在计算机实验室里实现的方法。与此相关，我们来简单谈谈模型调整。那么。实际上，我们上周就讨论过这个问题，当时我们讨论了有些模型会过分强调模式。因此，如果不注意调整，它们就会对数据中的不同模式过于敏感。我们所说的调整，基本上是指对任何参数的选择。例如，你的 lambda 就是 lasso 回归中的一个调整参数，因为它是一个你可以移动和改变并尝试不同值的参数，这对你的参数、你的模型以及你的模型与数据的契合度和灵活性都有影响。因此，我们会在系列讲座中多次讨论这个问题，这取决于你有什么样的模型。你有或多或少的调整参数。因此，你对模型施加影响的机会或多或少，或好或坏。上次我们还简单谈到了偏差和方差这两个统计概念。因此，如果我们知道我们的自变量之间的真实关系被描述为某种模型，那么这里的 X 的 F 可以是，例如，我们之前在线性回归中谈到的线性组合。因此，这是任何类型的模型和任何类型的误差，用来描述你的偏差和结果，那么我们显然想要找到某种模型。我们想要估计某种模型 F hat，以近似 x 的真实函数 F，这是一种完美的关系描述。因此，我们要估算出能代表这种关系的最佳模型。这实际上是一个非常重要的概念，我们要明白，模型只是我们试图描述的现实的最佳拟合。因此，任何模型都不可能完美地描述某种情况。我们只是试图在现有数据和计算成本等其他限制条件下找到最佳模型。因此，你已经知道，我们希望通过优化或最小化误差来实现这一目标，而误差就是我们的估计值与这两个值之间的偏差。现在我们还没有谈到，但你可能已经听说过的一点是，我们通常首先使用训练数据来计算误差。因此，我们将数据集分成不同的部分。例如，其中一部分用于估计线性回归的较佳值。然后，我们真正感兴趣的是，这个模型在其他数据上的表现如何，而这些数据是它目前还没有见过的。因此，当我们试图预测某件事情时，我们显然不能用未来发生的事情来训练模型。因此，我们要知道模型在未见过的测试数据上的表现如何。这就是我们在训练数据上计算典型误差的原因。那么我们的模型在我们知道它应该拟合的数据上拟合得如何？因为这就是我们正在训练它的数据。这就是我们向模型展示的数据。但我们也对测试误差感兴趣。模型在适应一个它从未见过的新情况时表现如何？通常，我们会计算这些误差的平均值，或者同时报告训练数据和测试数据的两个值。因此，我们总是对其中的两个误差感兴趣，因为一个是描述我们的拟合程度，另一个是描述我们未来的改革程度。是的，这是一种有点复杂的解释偏差和方差的方法。偏差是指我们是否真正预测到了我们所说的预测结果。因此，我们的模型在发现我们试图解释的关系方面的实际表现如何？然后，方差是指该模型的可靠性如何，例如，在未来，或者如果我们展示它，如果我们多次应用它。因此，我们在预测中发现了多少差异？偏差基本上是指我们实际找到我们试图预测的东西的程度，而方差是指如果我们想要多次找到真实值，我们找到真实值的可能性有多大？因此，这两点都非常重要，因为我们显然要同时做到这两点，对吗？例如，我们正试图预测违约客户。同时，我们也希望能够用同一个模型反复预测违约和客户。因此，我们希望方差小，与真实违约偏差小，偏差小。我们实际上是在预测违约。显然，由于生活中没有任何事情是完美无缺的，因此通常不可能做到两者都一样好、一样完美。因此，我们通常试图在这两者之间取得平衡，这就是我们所说的偏差方差权衡（bias variance Trade-off）。因此，它们总是能准确地找到中间点，但如果你给它们展示一些它们以前从未见过的新东西，它们就会突然不能很好地表现出来，因为它们太专注于此，以至于无法精确地重复完全相同的动作。同样，高偏差增强算法也能生成更简单的模型，因为它们能更好地理解概念，并很好地描述我们要做的事情。但我们可能会对训练数据拟合不足。因此，我们在建模过程中存在两种模糊认识。所以，我希望你们能从中学到的是，数据的过拟合和欠拟合总是与偏差和方差的权衡有关。因此，我们是否与训练数据拟合得过于紧密，导致我们无法预测未来的任何其他数据，或者我们是否拟合得过于松散，导致我们无法预测和很好地预测我们所拥有的任何数值。我认为这与我们上周讨论的寻找数据的最佳拟合线非常相似。因此，你可以看到左侧这里，我们称之为拟合模型不足。因此，你可以看到数据的真实关系是这样的，但我们只是对其进行了明显的线性回归拟合，这是一条直线，我们并没有真正捕捉到其中的任何信息。是这样吗？就好像我们在这里捕捉到了一点，但这并不是一个很好的拟合模型。右侧是拟合度较低的模型，这是另一个极端。所以我们这里的模型拟合过度了。我们在跟踪数据中每一个微小的变化。如果我们尝试将这个模型拟合到其他数据中，这些数据也有类似的行为，但并没有在这里或下面出现新的点，那么这个模型就无法捕捉到这些点，因为它们离这个过度拟合的模型太远了。因此，我们真正要寻找的是介于两者之间的模型。它并不完美，但已经相当不错了。因此，我们得到了这种关系的一般非线性，而这正是这些过于简化的模型和这些过于复杂的模型之间的完美折衷。因此，一般来说，更灵活的模型意味着我们有更高的方差和更低的偏差。反之，如果采用严格的模型，则方差较小，偏差较大。而这两个量的相对变化，相对变化率，决定了我们的测试误差是增加还是减少。原因在于，如果我们在这类数据上训练模型，那么如果我们向模型中添加新数据，会发生什么呢？我们是否还能预测突然出现的新数据点？这幅图很奇怪。我不记得把它放进去过。好吧，这和我刚才说的很相似。我们正在绘制模型的灵活性曲线。这基本上就是第一句话。灵活性越高，方差越大。这就是橙色曲线，而偏差会减小。这就是蓝色曲线。然后，我们要寻找的基本上就是这个最佳点。这就是测试数据中误差最小的地方，也就是红线。好的嗯，是的，我们在最后几分钟讨论一下数据分割，然后下次再讨论采样。我刚才谈到训练数据和测试数据是两个概念。鉴于这种权衡，我们已经看到了如何在两者之间实现最佳权衡，那就是我们将数据分成两部分，一部分用于训练模型，以估算出更好的参数。而另一部分数据则用来测试模型，看看它在未见过的数据中表现如何，例如是欠拟合还是过拟合。因此，训练数据用于建立和调整模型。然后我们使用测试数据来评估模型的性能。因此，有几种更现代的方法实际上是多次分割数据。我们在讨论交叉验证和重采样等问题时会看到这一点，因为有时这有助于我们更好地适应训练数据和测试数据。例如，如果数据中某些类别的代表性不足，或者有时我们的数据在不同地点之间存在较大差异，那么这种情况就尤其明显。因此，如果数据中存在某种模式或结构，通常会改善你的模型。如果你进行多次重新采样，并分成多次训练和测试。因此，我们通常会将数据分成 70% 的训练数据和 30% 的测试数据。这是大多数人遵循的经验法则。这就是经验法则。所以，你可以选择任何你喜欢的数值。有人用 6040，有人用 5050，有人用 8020。这其实并不重要。关键是你需要的训练数据通常要多于测试数据，因为你想在这部分数据上准确地训练你的模型，然后再在数据的较小部分上对其进行评估，以检查并报告其中的误差。有些模型还需要额外的验证集。这样，你就有了用于训练模型的训练数据，然后你就有了超参数，例如，你想要调整和优化得更强一些的神经网络。因此，你有另一个数据集，然后在最后对测试数据进行测试。这只是相同逻辑的不同方法。底层用于更复杂的模型。例如，神经网络通常使用底部。如果使用线性逻辑回归决策树，则通常使用顶部。因此，训练数据和测试数据通常非常非常重要。我从来不希望看到有人这么做。如果你向模型展示了测试数据，那你就大功告成了。之后你就不会再做任何调整、重新运行或实验了。原因是一旦测试数据被看到，它就是模型中的信息，你就不能再称之为独立的样本外测试数据了。只要他一出现，你就已经在模型的任何步骤中使用了它。因此，一旦你对测试数据进行了某种调整，它就不再是测试数据了。它变成了训练数据。好吗？所以千万不要在测试数据上做任何优化。最简单的方法是我们将随机样本分成训练集和测试集。测试验证集。我想说的是测试集，在某些情况下，某些类别的频率较低，我们会尝试复制这种分布，例如，对代表性不足的类别进行超采样。我们稍后会讨论这个问题。现在我们要讨论的一个有趣的问题是重采样。交叉验证和重采样技术中最有名的方法是 K 折叠交叉验证。其原理是将数据集随机划分为大小相等的 k 组。例如，如果你有 100 个数据点，而 K 值设置为 10，那么你就会有十个数据集，每个数据集有十个数据点。然后，除了最后一个数据集之外，你可以将训练数据与所有数据集进行匹配。因此，这十组数据中有九组会成为你的训练数据。最后一组十个元素将成为你的测试数据。