

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。

I.大家早上好大家可能这张幻灯片是上周的幻灯片。实际上，我之所以再次播放这张幻灯片，是因为我收到了一些关于 PCA 的问题，尤其是 PCA 与回归的结合，我们称之为主成分回归。因此，我想再花两三分钟时间，再次解释一下 PCA 和 PCR 背后的理念，以确保其绝对清晰。我希望大家还记得 PCA 是一种降维技术。因此，PCA 背后的理念是，如果你的数据集中有很多特征、很多变量，而你又不完全确定如何才能最好地将它们全部纳入模型，那么实际上减少空间的一种方法就是使用 PCA，基本上将多个变量组合成分量，这样就不会再有 50 个变量来解释数据集中的不同概念或特征，而是减少到特定数量的分量，比如说 10 个。然后，这十个组成部分就是原来 50 个特征的组合。这些成分的组成方式应能包含数据集中的大部分方差或大部分信息。我想这一点相对来说不言自明。我们的想法是，既要保留尽可能多的信息，又要减少解释这些信息所需的变量数量。这背后的主要想法，以及为什么它能起作用，就是在很多数据空间中，我们实际上存在共线性。因此，不同变量之间存在相关性。就信息而言，这意味着它们部分解释了相似或相同的信息。因此，我举了这样一个例子：你有一份问卷，问卷中的不同问题实际上测量的是非常相似的概念或类似的潜在想法或特征。如果是这样的话，这些问题就会相互关联。因此，这些数值实际上会彼此强烈相关，因为它们测量的基本上是同一件事。而 PCA 就是利用这种共线性。然后将这些相关变量的特征组合成它们的成分。这也意味着，根据定义，PCA 的成分是相互独立的。因此，特征之间或 PCA 结果的成分之间不存在共线性，这对线性回归非常有用。我想这也是 PCR 成为独立方法的原因之一。因此，从理论上讲，PCA 本身只是一个免费的处理工具。你可以将其与任何方法结合使用。你可以只做 PCA。然后你就有了你的成分，你就可以把它们作为变量用于任何类型的模型、决策树或我不知道的英国和 N 任何你感兴趣的地方。PCR 作为一种独立的方法有点。这有点令人费解，为什么我们要这样描述呢？是的如果你能关闭它。谢谢。当我们把它描述为一种独立的方法时，尽管它基本上只是 PCA 与回归的结合，但我认为原因有二：其一，回归中的一个主要问题是，如果你有相关的特征，相关的变量。因此，有时你会尝试在 Python 中运行线性回归。Python 基本上会给你一条错误信息，告诉你，你知道，你的数据集中存在共线性，不能进行线性回归。然后，你必须弄清楚，哪些是相关特征。然后你可以尝试移除其中的一些特征，等等等等。实现这一目标的另一种方法是将 PCA 与回归相结合。因为正如我所提到的，根据定义，各成分之间互不相关。因此，这将完全消除回归模型中的共线性问题。现在，PCR 作为一种方法，在选择成分数量时又向前迈进了一步。正如我前面提到的，你希望选择的成分能够包含数据集中的大部分信息或变异。因此，这就是选择数据中组件数量的方法。例如，你希望捕获数据集中 70% 的信息。这是你可以选择的经验法则。因此，这将成为 PCA 解决方案的评估参数。因此，你可以根据需要选择尽可能多的成分来实现这一评估。PCR 中捕捉的信息方差参数。要知道，在回归中，我们可以使用误差信息误差度量作为评估线性回归的工具。因此，PCR 基本上就是利用回归中的误差信息这一评估指标作为我们对成分的评估工具。因此，我们要选择能优化线性回归的成分数量。这就是为什么它基本上是一种组合方法。因为我们不是先做 PCA，然后再做线性回归，而是在回归模型中优化成分的数量。你在计算机实验室里已经看到了，我们使用交叉验证来选择成分的数量。但实际上，我们是在线性回归的基础上进行交叉验证。因此，我们使用不同数量的成分进行多次线性回归。然后，我们选择能优化线性回归的成分数量。这就是 PCA 和 PCR 背后的理念。对此还有什么问题吗？或者比上周更清楚了？很好。这需要一点思考，但基本上是 PCA 预处理、组合特征和 PCR 在线性回归中使用这些组合特征。因此，在这种情况下，线性回归公式就是利用特征和成分来预测 y。每个成分都是一个变量。例如，贝塔零加上贝塔一乘以 PCA 一，再加上贝塔二乘以 PCA 二。这些就是你的成分。在这种情况下，你只需将它们视为变量，因为我们的想法是，它们是原始变量的线性组合。因此，你可以将它们视为正常变量，但需要注意的是，由于它们是线性组合，你不太容易解释它们。因此，难以解释的问题会使线性回归的可解释性大打折扣，但却消除了共线性问题。所以非常有用。以上就是对 PCA 和 PCR 的简单介绍。PCA 和 PCR在今天的讲座中，我们将继续 "我没有将边缘设置为默认"。你疯了吗？糟糕的浏览器来吧好了，开始吧这就是当讲师而不是老师的好处我可以告诉你我对事情的看法好了，现在是第九周你们快完成了快结束了不管是好是坏接下来的几周我们基本上每周讨论一组或一种特定的方法。因此，这周我们将讨论三种最佳方法决策树，以及基于决策树的方法。下周我们将讨论支持向量机。然后我们将在第 11 周讨论神经网络。然后你就大功告成了。所以，你差不多已经学完了。是的。正如我上周提到的，我们讨论了 K 和 N。我们还讨论了通过 PCA 和 PCR 进行特征选择和降维。本周我们要讨论的是基于决策规则的分类器。具体来说，我们将讨论决策树，包括分类回归树。我们将研究如何将这些树组合成集合，而不是单独使用。这就引出了一般的袋集方法。我们还将谈谈提升树。我们还将讨论一下随机森林，这是每个人都喜欢的方法。如果它不是神经网络的话。这三种基本方法非常有趣，因为它们背后的理念非常直观。从数学角度来看，它们的基本原理是分层。它们将预测空间划分为不同的区域。这听起来有点奇怪，但我们很快就会看到它在图表中的实际效果。这样做的目的是，如果我们想预测一个新的点，一个新的数据点，或者一个我们还没有类别标签或某种输出测量的无标签数据点，我们就会使用预测空间的平均响应。因此，这与 k 和 n 的概念类似。我们正在定义一个空间，一个预测空间，或者我们正在定义相邻空间，然后根据已知的相似点、相近点、同一预测空间中的点的行为来标记新点和未标记点，诸如此类。因此，我们总是在考虑如何预测某件事情？我们正在使用以前的信息，我们已经掌握的过去的信息。然后在此基础上，我们寻找某种意义上的相似性，并以此进行预测。因此，决策树背后的理念就是，我们如何划分预测空间的这些规则可以用一棵树来概括，决策树也因此得名。决策树最棒的地方在于，它不仅仅是一棵树。实际上，它看起来就像一棵树。因此，你可以直观地看到预测背后的过程。稍后，我们还将研究更先进的方法，特别是多棵树的组合。你会发现，决策树非常神奇，因为你可以将其可视化，而且它们看起来非常漂亮，实施起来也非常直接，并能解释一切，但不幸的是，它们并不是真正的好方法。因此，在很多情况下，单棵树是一种很弱的方法，至少在很多情况下，如果你的数据比较复杂的话。因此，如果是一个非常简单、直接的模型，一个想法和数据空间，决策树就能做得足够好。不用担心这个问题。但如果复杂度更高，特征更多。例如，你有很多复杂的非线性关系。决策树本身并不能很好地完成任务。所以我们通常说，与其他方法相比，决策树的预测能力相对较弱。但实际上，将决策树结合在一起是非常非常强大的。因此，我们稍后将讨论随机森林。随机森林实际上被认为是最强大的预测工具之一。因此，如果你阅读科学论文，并将随机森林与不同的模型进行比较，在很多情况下，随机森林的表现都大大优于更复杂的模型，这有点出人意料，因为你会发现随机森林背后的想法相当愚蠢，但它确实有效。因此，在机器学习中，我们并不关心某些东西为什么有效，只要它有效，我们就高兴，我们就会使用它。让我们来看一个例子，看看它到底是什么样的。去年我用这个例子的时候，我不得不问在座的各位是否真的熟悉棒球，因为我一点都不熟悉。所以你们中的一些人，可能是来自美国的人，比我更了解棒球。我对棒球的了解完全来自动漫。所以这真的是歪打正着，不太准确。所以我们的想法是 这就是我们的数据我们根据棒球运动员的命中率 以及他们成为职业棒球运动员的年数 来计算他们的工资。你可以看到，工资被编码为蓝色或绿色。如果工资较低，则用黄色或红色编码；如果工资较高，则用黄色或红色编码。因此，我们用肉眼就可以看出，这个区域的红色和黄色较多。这个区域的蓝色和绿色多一些，但并不完全一致。如果你看一下底部，这里也有不少蓝色的点。因此，仅仅使用线性回归或类似的方法，画出一条直线，并不能真正捕捉到模型中的所有动态变化。那么，你会如何划分呢？你怎么看？有什么好的方法来划分这个空间？我可以看到手的动作。你觉得呢？你觉得呢？我可能会在这里画一条线 So I would probably say like draw a line through here.听起来不错吧？在这里画一条线这样基本上就有三个空格了比如说，看起来就像这样。这是决策树所能实现的，而线性回归则无法实现。所以。正如我所提到的，决策树基本上是利用多种决策规则和决策规则系统将数据空间划分为不同的区域。举例来说，在这种情况下，我们可以说有一面墙不到五年，每个不到五年的人似乎都有相对较低的薪水。除了底部的这种异常值，在本例中我们会很方便地忽略它们，因为它们只有两个，所以它们完全是输入错误的数据或非常奇怪的球员。我不知道怎么会有人能做到这一点，所以我们就忽略他们吧。方便起见，我们假设少于 5 年可能是其中一堵墙，然后也许在命中率变量上有另一条规则，做类似这样的事情，比如多于 100 或 110。这样，你就可以计算出这些完美的 "分界线 "值究竟在哪里，从而对空间进行最佳分割。你可以在右边看到结果。这就是我们称之为树的原因。现在，这看起来还不是很像树，因为这是一个非常简单的问题。但是，如果你有更多的变量，并且继续下去，它就会不断生长。它从顶部一直延伸到底部。因此，我们首先要划分变量年份的空间。例如，小于 4.5 年。然后我们再往下走。如果是小于 4.5 年，或者是大于 4.5 年，我们就已经在预测工资了。因此，在右侧，我们将进一步划分空间。因此，你可以看到，这背后是一个非常聪明的想法。有时很容易得到一个值。因此，如果小于 4.5 年，我们就可以得到所有这些值，并将它们分配给相同的工资 4.5，但在其他空间，情况就比较复杂了。因此，我们需要在空间的右侧制定另一条规则，在这种情况下，我们可以说，如果他们的命中率小于 117.5 次，不管这意味着什么，那么他们就会得到 6 次的工资分配。如果超过了这个数字，他们就会得到 6.74 的薪水。因此，这种逐步分割数据是决策树的核心原理。当然，我们也有一些行话，一些决策树使用的语言。我们通常将其分为内部节点和终端节点。在这种情况下，你有两个内部节点，即这里和这里。这就是决策树内部的分支。然后是终端节点，如果是一棵树，你也可以称其为树叶。这些节点就是这些数字。因此，树的最边缘被称为树叶。每片叶子上的数字就是落在那里的观测值的平均响应。因此，基本上，如果你有一个特定的观测点，比如这里的某个特定数据点，让我们以此为例。你想对它进行预测。那么你首先要看它是否有超过 4.5 年的时间。你会说是的。然后你可以说它是否超过 100 年。那 117 次点击呢？你会说没有。因此，对这个小点的预测值将大于 4.5，但小于 117.56，对这个特定数据点的预测值为 6。因此，对这些数据点中的每一个，你基本上都会在决策树中画出一条路线，看看你最终会走到哪一片叶子。有些树叶比其他树叶更早结束。没错，我们的决策树就是这样划分空间的。你可以看到，我们有三个决策空间。基本上就是三片叶子，每个数据点都有三个可能的值。我们有一个。底部有两个。顶部有三个。这些值基本上就是这些值。所以这就是我们的一这就是我们的一这就是我们的一。然后是小于 117这是底部。这就是我们的二然后这里是三所以，叶子的数量总是和你的决定空间相同。或者说是你预测的空间。因此，我们的区域被称为终端节点或叶子。没错，决策树通常是倒着画的。这就是决策树的主要区别。在真正的树上，它们是从天花板长出来的，也就是树底的叶子。从理论上讲，你也可以把树叶倒过来画，但在大多数语言中，从上到下是一种直观的阅读方式，所以这样画出位置树也是合理的。是的，我们一直都说，位置树的分叉点是指内部节点，或者我称之为分支。所以我觉得这样更直观。但大多数人会说是内部节点。这也是对图论的一种补充。因此，如果你熟悉基于图的方法和图论，那么你就会熟悉所有这些围绕节点和边的语言。如果我们的想法是，这些区域是不同的、不重叠的。我认为这也是一个非常重要的区别。因此，我认为我们在讨论聚类时已经非常、非常简要地提到了这一点。但实际上，我们讨论的大多数方法都是在描述不同的群体。所以，就像有一个或另一个标签给你。我们不会给任何数据点分配混合标签。我们在做决定。有些方法我们称之为模糊法。例如，有一种模糊聚类算法，我们不给某个点指定特定的组或标签。实际上，我们会在决策树中为多个组分配成员度。我们并没有真正这么做。我相信有人发明了模糊决策树。有人发明了一切。但在这种情况下，假设我们所有的分组都是不同的，没有重叠。因此，我不会只在一片叶子中对你进行降级。每个点都属于一片叶子。是的，对于每一个属于该组的观测点，我们都会根据训练中反应值的平均值进行预测。这样就说得通了。我们先用训练数据构建树，然后再输入测试数据，最后再看数据的最终结果。是的，从理论上讲，区域可以有任何形状。因此，我们通常会按照这些矩形框来画，因为这是最简单易懂的，尤其是在二维空间中。这就是我们画矩形框的原因。但决策树实际上也可以寻找非线性关系。因此，你可以寻找非常灵活的空间，因为它不一定是一个完美的方框。出于简单性考虑，它大多是一个方框，因为它最容易计算，最容易绘制。我们究竟该如何找到这些盒子呢？我们又回到了误差最小化的问题上。你应该记得 RSS 是我们的残差平方和。这就是你的误差信息。你可以计算出每个方框的误差。在这里，你可以查看每个值。它与预测值的实际偏差有多大，预测值是该空间中训练观测值的平均响应。因此，我们的每个方框只有一个值，而对于其中一些值、其中一些点，其偏差会比其他值更大。我想这一点是比较清楚的。因此，矩形框中每个棒球运动员的薪水是不一样的。虽然会有偏差，但矩形框的画法是将偏差最小化。换句话说，每个方框内的实际值和预测值的偏差应该是最小的，这基本上就是误差。是的，这是另一件事。考虑所有可能的分区在计算上是不可行的，因为有很多不同的分区，对吧？所以你可以在很多不同的地方画线，用线来划分空间。对，用来划分空间。因此，我们通常采用一种叫做递归二进制分割的方法。我们从顶部的第一个内部节点开始。然后我们会想，在这个特定的时间点上，什么是最好的分割？在这个特定的时间点我们忽略未来。我们忽略过去。我们只是想，好吧，如果这是我们的空间，我们应该在哪里划线，才能真正。在这个特定的时间点上，将误差最小化。我们称这种方法为贪婪法，因为我们只考虑这一步。我们不会考虑是否应该先在不同的变量上进行划分，或者添加一个不同的值，或者其他类似的东西。我们只是说，好吧，第一个值是什么？我们先看作为棒球运动员所花费的时间，然后再看该球员的安打数。举个例子，我们不考虑未来，我们只考虑现在。嗯。老实说，这可能是件坏事，因为这可能意味着，如果你不考虑未来哪一步会更好，实际上，你往往会在同一个变量上分裂多次，例如，你可能会在一个变量上分裂一次。因此，你可能在很早的时候就对一个变量进行了一次拆分，然后在另一个分支中，又对同一个变量进行了拆分，之后又在另一个分支中做了一次。因此，这种方法有点无效，但却很有效，因为你不考虑历史，也不考虑未来，所以做起来非常快。只考虑在这个特定的时间点上做出最好的决定。那么，这看起来像什么呢？实际上，我们有一个漂亮的五步流程，我们首先选择一个预测因子，例如花费的时间。然后选择一个切点，我想我们说过是 4.7 年之类的。这样，预测空间的分割就能最大程度地减少误差。这就是你刚才说的。我们要以这样一种方式分割空间，使每个空间中的点都最接近或尽可能接近该空间中的预测值，然后我们再做同样的处理。因此，我们会想，好吧，下一步应该取哪个变量。我们下一步应该取哪个变量的值。然后再一步步拆分。我们总是试图将实际值和预测值之间的误差降到最低。是的，很明显，我们不是分割整个空间，而是继续分割我们现有的空间。因此，我们首先按照年龄进行分割，然后我们就有了，例如，我们的新区域。这里。如果我们先从上到下进行分割，那么我们就会想，好吧，这里有两个空间，左边和右边。我们该如何更好地分割它们呢？哦，那就分割右边的吧，我们从中间分割右边的。就这样，我们一步步递归地分割空间。同样，我们要尽量减少 RSS。然后一步一步继续这个过程，直到达到某种停止标准。比如说，你可以继续下去，直到没有一个区域的观测值超过五个。因此，有不同的方法来决定何时停止。这实际上取决于你想走多远。事实上，决策树的一大危险在于，它们往往会很快、很强烈地过度拟合数据，所以决策树绝对容易过度拟合。你可以使用停止标准。例如，你究竟想通过下一次拆分减少多少时间来决定，好吧，减少的时间不够值得，或者空间变得太小了。我想停下来。我想停了。我们总是说，我们要对测试观测的每个响应进行预测。然后使用该区域内训练观测数据的平均值。所以我们认为同一区域内的点是相似的因此，我们将所有的点放在一起进行投票。我们的测试值应该是多少？我们刚刚说过，树很容易过度拟合，因此有不同的方法来避免过度拟合。我们说了停止标准，然后开始检查每片叶子上有多少个点。但实现这一目标的另一种方法是对树进行修剪。所以没错，我们都在用这些园艺术语来形容决策树。这很可爱。所以我们说，一棵更小、分裂更少的树可能会带来更低的方差和更好的解释，但代价是会有一点偏差。也就是我们说的方差偏差权衡。因此，我们希望避免过度拟合训练数据，即使这意味着我们在训练数据上的准确性会稍差一些，但至少我们可以在未来对测试数据进行准确预测。因此，举例来说，你只能让树生长这么长的时间，直到达到某种太小的下降幅度，但没错，这是非常正确的。这有点目光短浅，因为我们已经说过，每一个分割都只是对一个分割进行思考，而不是对一个特征和过去进行思考。因此，可能很快就会有一个非常非常好的分拆，这对减产来说是非常好的。但由于它提前了两步，你看不到它，所以在这种情况下，你基本上过早地在你的空间里停止了。因此，在这种情况下，你可以做的另一个选择就是种植一棵非常非常大的树。这样一路走下去，过度拟合你的数据，然后再把它修剪回来。这又回到了图论的概念，即基于图的模型。如果你在本科阶段学过图论，你可能会知道，你基本上是在寻找最弱的边或最弱的链接，然后进行修剪。因此，这里的想法是，你要看不同的树。而你要找的是在这些树中最小的一棵树，它仍然做得很好。所以，你基本上是在想：好吧，这棵树在所有这些方向上都在生长，但我只对这里的空间感兴趣。这有点过度拟合，做得很奇怪。因此，我们只对决策树的子树感兴趣，我们要剪掉其他边缘，因为这些边缘不太好。这里的想法是，你可以使用一个调整参数。你可以使用交叉验证来选择剪枝或调整参数的最佳值。然后，你就能真正找到最佳子树，它仍能为你提供一个非常好的 RSS 值。因此，一个非常好的减少误差的值，但复杂度要低一些，而且比这个值要小一些。没错。快速总结。我认为决策树比较容易理解。这就是为什么我说我喜欢教决策树，因为它们并不难。比如在这种情况下，我们建议你使用递归二进制拆分法。请记住，这是一种逐步拆分的方法，总是分成两个区域，只看可能的最佳步骤。听着，在你所有的数据和训练数据上建立一棵非常非常大的树。当每片树叶都有一定数量的最小观测值时，你就可以停止了。我想我们之前说过，5 个取决于数据集到底有多大。然后，你会使用某种成本复杂度剪枝法，即在阿尔法函数的作用下，你会发现一个子树仍然做得很好。使用交叉验证来选择参数 alpha。等等等等。这样，你就能得到一棵实际表现仍然很好的子树，但它比原始树更小，因此更不可能过拟合。所以修剪树相对简单，就像你基本上。你以前见过这个。这么说吧。在我们讨论回归正则化的时候，你也见过类似的方法。所以，如果你回想一下你的 lasso 回归，举个例子，我们也有这个调整参数，基本上我们必须根据模型的复杂度来调整。在线性回归中，模型的复杂度就是预测因子的数量或变量的数量。因此，你希望减少回归中的变量数量。而在这里，你基本上想要做的是树的大小。你想缩小回归树的规模，同时优化它对数据的拟合程度。你想优化误差，缩小回归树。所以从根本上说，剪枝与回归中的正则化非常相似。因此，这些概念彼此非常相似。下面是我们棒球示例的一些精美图片。我不知道为什么会有棒球的例子。老实说，我觉得我应该换个语境，因为我根本不知道自己在说什么。好了，左边的这些变量我无法解释，但你们应该还记得，这三个变量中基本上有一部分就是我们之前的变量，对吧？我们把今年的这个内部注释拆分到这里，把这个内部节点拆分到这里。但现在，我们实际上已经进一步构建了这棵树。因此，我们做了之前说过的构建一棵大树的事情。你可以看到，我们有比这里更多的分叉。我不知道 RB 是什么。RB是什么？咦？就像在你之后我们有点去那里。啊，所以它的运行时间。我的意思是，这是一个非常糟糕的例子 在这种情况下，是不是？因为有些球员在某些方面比其他人强有意思好吧是的，这就是我想说的我觉得这张图很好 你可以再看下这张图的年份没错所以你可以看到，在同一棵树上，变量实际上被使用了很多次。这也是决策树有时效率不高的原因，因为基本上你总是在想，什么才是最好的变量。而这个变量可能会被多次使用。这说明了什么？通常情况下，如果一个变量被多次提及，那么它就是一个非常好的预测变量。所以模型非常喜欢它。所以，这就是为什么它会多次出现。你也可以在我们的棒球示例和散点图中看到，在我们的图中，工资也是一个很好的预测变量。因此，这就是它多次分裂的原因。有时，即使是相同的值，你也可以看到这里小于 3.5。年数也小于 3.5。因此，它实际上是在进行相同的分割，但分割的组别略有不同。这就是为什么它的效率有点低，但在我们这个相对简单的例子中，它做得足够好，在右侧，你还可以看到树的大小是如何增长的，误差实际上也有所改善。因此，你可以看到树的大小就是内部节点的数量。你还可以看到，在所有不同的数据集上，这棵树是如何首先降低误差的。这就是训练交叉验证和测试数据。然后，实际上我们在测试数据和交叉验证数据上的误差又有所增加。这就是过度拟合。可视化。你可以看到，训练数据的误差仍在减少。因此，我们仍在提高树与训练数据的拟合度。但实际上，我们在测试数据和交叉验证数据上的误差。因此，我们所说的交叉验证基本上是指，我们将数据集分割开来，在多个子数据集上训练模型。然后比较，这就是我们的交叉验证误差。因此，你可以看到这个误差实际上在增加。你所报告的测试数据实际上也在增加，因为它过度拟合了训练数据。所以你可能会说，比如这里就是我们的甜蜜点。在这里，测试数据的误差实际上是最小的。你可能会说，这个误差同样很小，但它的复杂度确实在增加。而且交叉验证误差也非常大。所以要小心这种树。我认为我们要寻找的最佳点是年数、RBI 和命中率这三个分值。然后就大功告成了。所以，这就是你要寻找的一种计算效率高、工作足够出色的小树。我们刚才说的是回归树。我们预测的是某种连续的数字结果，比如工资。但显然，你也可以将回归树用于分类任务。它们的工作原理与此非常相似。因此，在这种情况下，我们要做的就是预测每个观察结果属于某个特定类别，特别是训练数据中最常出现的类别。在该训练数据中，在特定的预测区域内。从理论上讲，你将空间划分为不同的子空间。你预测空间。然后，你会寻找所有这些数据点的投票情况。它们属于哪一类。你在该空间中的测试数据点也应该属于这一类。因为它们相似，彼此接近。我们都在投票。所以这基本上是一回事。分类树和回归树显然有一些不同之处，比如，我们不能使用 RSS，不能用误差来衡量分类树的质量。因为你不能用类标签来计算误差。从这个意义上说，类标签是没有意义的。因此，我们需要的是分类错误率。这与我们在逻辑回归中所说的非常相似。因此，基本上是寻找该区域内不属于最常见类别的训练观测值的比例。例如，如果你有 20 个点，它们都在投票，其中 15 个点投红色，5 个点投蓝色。那么预测结果就是红色。但你还是想知道其中有多少人的投票结果是不同的。因此，这基本上就是你的分类错误率。而在实践中，这种方法并不能很好地实现 "诡计增长"。因此，我们实际使用的通常是基尼指数或节点纯度指数。在这里，我们仍然在寻找一个非常相似的概念。因此，你仍然可以看到我们之前说过的 P hat MK 是第 m 个区域中来自 K 类的训练观测值的比例。所以我们仍然在寻找这个值。因此，我们仍然对这些点在我们的空间中的投票情况感兴趣。但现在，我们正在寻找所有不同类别的总方差。因此，我们基本上是在优化每个点的均匀性或一致性。这些点如何始终如一地投票选择它们应该属于的正确类别？那么，这些音符的纯度有多高？如果有 20 个节点投票，其中 15 个投红色，5 个投蓝色，那么投票结果的纯度如何？我们的想法是基尼指数越高，我们的节点在投票决定中就越团结。是的，这听起来很像政治投票中的投票。我知道这很奇怪。现在我们来看看基尼指数，它是一种交叉熵。唯一不同的是，这里基本上是取对数。所以你可以看到，从计算方法上看，它真的很相似。你感兴趣的仍然是每个类别中的每个点的投票情况。现在，你可以取其对数。你还是要把所有可能的类别加起来。交叉熵为负出乎所有人意料的是，如果你看一下类似的计算结果，它们在数值上是相似的。因此，如果你计算基尼系数和交叉熵，它们通常会告诉你同样的事情。因此，它们都会在特定节点的节点纯度上达成某种一致，至于你选择哪一个，完全取决于你自己。因此，两者之间并没有什么大的区别。这只是一个偏好问题，而且在大多数情况下，它们都会达成一致，所以这并不重要。在这里，我们实际上有一个例子，在这种情况下，不是棒球，而是旅游。因此，这与我的实际工作更接近一些。显然，我们有五位游客来到爱丁堡。所以我们要问所有人，他们参观过城堡吗？有没有买苏格兰短裙？买了多少瓶威士忌？我们要预测的是他们是不是游客。因此，这些显然都是非常旅游化的行为，而我们感兴趣的是预测他们是否是游客。因此，我们现在要做的基本上就是计算基尼指数或两个类别的节点纯度。例如，在这种情况下，我们有两个类别，要么是游客，要么不是游客，这就是我们的最后一列。然后我们可以计算这两个类别的基尼指数。因此，游客类的基尼系数是 3 比 5。而非游客类别的基尼指数则是五分之二。因此，我们就可以计算出拆分前的基尼指数。因此，目前我们所有的数据都已拆分，我们计算的基尼系数是游客类节点纯度的 2 倍。非游客没有纯度。这样，在拆分之前，整个数据集的基尼系数为 0.48。我们这样做的原因是，记住我们要做的是以这样一种方式进行拆分，以提高我们的基尼系数。因此，我们需要某种基线值。这就是我们的基准值。现在，我们要想好应该对哪些变量进行拆分。那么，第一个要分割成两个空间的变量是什么呢？例如，我们先检查变量是船还是不是船？是的这就是变量B，既有被杀的，也有没买被杀的。基本上，你要做的就是再次查找所有游客，例如，也买了东西的游客，以及所有没有买东西的非游客。然后再举例说明两者之间的区别。例如，他是一位没有购买短裙的游客。这就是一个例子。没错。这基本上是在你的笔记中创造了非纯粹性，因为你这里有两个游客。所以他们是三和四，但他们有不同的行为。因此，在这种情况下，我们可能会说，好吧，也许这是一种影响我们如何能分裂的变量。这就是为什么我们要计算基尼系数。因此，我们有我们的 P t 游客。我们计算的是游客和非游客乘以 2。然后我们对非游客的基尼指数做同样的计算。因此，我们既买了东西，又不是游客，既没买东西，也不是游客。这样，我们的基尼指数就是 0.266。因此，我们的基尼指数从 0.4820.266 降到了 0.4820.266，这意味着，通过对基尼指数进行拆分，我们基本上可以改善我们的节点纯度。因此，你可以将其与我们的不同变量进行比较，哪个变量给你带来的改善最大。基尼系数降低幅度最大的变量，就是你在那个特定时间点上进行拆分的变量。因此，请记住我们之前说过的话。这种分割是贪婪的。所以我们每次只检查其中的一个变量。然后我们忽略过去，忽略未来。我们只是在寻找目前基尼系数降低的最佳方案。是的，这基本上就是你如何为一个数值做同样的事情。因此，在这种情况下，你实际上可以使用 RSS，因为购买的威士忌酒瓶实际上也可以使用 RSS。我就简单跳过这个，因为我觉得你会看起来很累。我们先休息十分钟，然后回来继续讨论决策树。好了，让我们结束关于决策树的讨论吧。在休息之前，我们花了几分钟来思考如何选择最佳变量进行拆分。所以，如果你回想一下这个树的例子，基本上我们要找出的是，我们应该选择哪个变量来进行实际的拆分。而最好的分割变量，就是最能分割我们的空间的变量。所以，你应该还记得我们一开始举过的这个例子，我们说，好吧，年龄似乎是一个非常好的分割变量，因为它定义了这两个行为非常相似的群体。我们在这里也尝试做了同样的事情。唯一不同的是，我们对模型性能的衡量方式不同。因此，我们并不是要预测薪资。我们预测的只是一个标签。这就是为什么我们的计算方法有所不同。但原理还是一样的。因此，我们仍在努力创造所谓的纯节点。当我们说纯节点时，我们的意思是，基本上我们希望所有游客都在一片叶子里，而所有非游客都在一片叶子里。这样才是完美的分割。因此，如果你有这样的分裂，例如，如果我们说两个人都被杀了，让我们把两个人都被杀了想象成一个完美的变量。那么所有游客都会购买 "被杀"，而非游客则不会购买 "被杀"。因此，这将使它成为一个完美的分割变量，因为如果你看他们是否都被杀，就能完美地预测他们是否是游客。但实际上并不是这样。因此，没有一个分割变量是完美的。因此，我们计算基尼系数，以此来衡量其他分组的纯度。因此，拆分的实际效果如何。这就是为什么我们要研究这些比例。游客人数和非游客人数。然后再将其分成两部分。我们用基尼指数来计算，因为我们无法真正计算出一个标签的误差。买没买短裙是一个二元标签。因此，我们无法计算其误差。但如果我们有一个数字变量，我们就可以计算。例如，购买威士忌的瓶数就是一个数值变量。如果有了这些变量，就可以计算出误差。在这种情况下，误差基本上就是购买威士忌的瓶数对游客与否的预测程度。举例来说，游客比非游客购买更多的威士忌。因此，您需要查看游客购买的瓶数和非游客购买的瓶数，并最大限度地缩小两者之间的差异。举个数字例子。你可以在顶部看到我们的计算结果。例如，我们把瓶子分成两个值。因此，算法实际上会选择一个最佳的分割值。比方说，我们选择的值是 2。我们对某人是否买了两瓶以上的威士忌感兴趣，以便预测他是否是游客。因此，如果我们进行拆分，这是一个二元拆分。因此，要么超过两瓶，要么不超过两瓶，我们可以看到有两个结果区域。你还记得吗，这就是把我们的空间分成了两个区域。我们将计算出在该组中购买的威士忌酒瓶的中值或平均值。因此，在这种情况下，比如说，我们有一号空间，也就是一号和二号访客。让我检查一下。给哈哈哈哈真有意思啊，我们走吧。现在 Now.我比以前聪明多了游客二和游客五买的威士忌少于两瓶 So we have visitor two and visitor five are buying less or buying less than two bottles of whisky.而访客一、三和四 购买的威士忌超过两瓶我们就是这样划分的。就这样这是第一组二号和五号游客。他们分别买了一瓶和两瓶。第二组是2号，分别是1号、3号和4号游客。他们分别买了四瓶、三瓶和四瓶。这就是我们对两个空间的划分。然后我们计算出每个空间的平均购买瓶数。然后，基本上就可以计算出每个空格的误差。如果有人买了一瓶，那么他与预测值的误差是多少。因此，这里的两位游客实际上只买了一瓶威士忌。但我们预测的是 1.5 瓶威士忌。所以误差是 1-1.5 平方。下一位游客是五号游客。他们买了两瓶威士忌。我们本来预测是 1.5 瓶。因此，误差为 2 -1.5 平方。因此，所有这些基本上都是我们在预测这些不同的人是否购买了多少瓶威士忌时的误差。在进行任何形式的拆分之前，我们先进行大于 2 的拆分。对于 2 的不同值，我们也会做同样的处理。因此，这里以瓶子的不同值为例。天哪，好吧。对于每瓶多少瓶的值。因此，举例来说，如果我们按照大于 3 的瓶数进行拆分，我们也会做同样的事情。我们会看有多少人买了三瓶以上，多少人买了三瓶以下。因此，现在实际上只有两个人买了多于三瓶的酒，三个人买了少于三瓶的酒。三个人买的少于三瓶。我们仍然可以计算出预测他们实际购买数量的误差。我们可以看到这里的误差是 2。一小时前的误差是 1.167。因此，将瓶数分成两瓶以上会更好，因为我们对该组购买瓶数的预测误差会更小。因此，我想从这里得到的重要启示是：基于树的方法。基于树的方法。基本上，我们有回归树和分类树，它们的工作原理非常相似，在划分的空间中，它们通过最常出现的值进行预测，或者通过测试数据在特定空间中的预测平均值进行预测。它们之间的主要区别在于如何找到最佳分割。因为这取决于你是要预测不同的类别，还是要预测特定的值。此外，它们还取决于变量的编码方式。因此，我们已经看到了数字变量，我们可以使用减少误差的方法。而我们看到的分类或类别标签分类变量，则会使用基尼指数（Gini index）来减少误差。但这两者的基本原理非常相似。因此，我们将有几分钟的时间。今天就五分钟吧，因为也不算太多。接下来我确实想谈谈乞讨，但我们还是要快速讨论一下决策树，讨论一下你目前对决策树的了解。然后，我们将讨论它们，并收集你们所做的笔记。所以，在讨论过程中也要记得做笔记，因为这显然不在幻灯片上。你们可能会发现这很有用。很好。但是你。认为。我得到。这个我得到这个。好了 Okay.我们准备好了吗？谁想今天开始？谁想开始？这是一个非常。直观的可视化为您服务。这个节目充满了文字不，我发明了它。可以可视化。这就对了是的，没错。我们还谈到了 k 和 n 的直观方法，以及线性回归。所以我认为决策树的优势之一在于，你可以很容易地向大多数人解释它们，至少是那种基本原理。它基本上就是一组决策规则。因此，你只需设定一组二元决策，要么大于这个值，要么小于这个值，要么是这个值，要么不是那个值。然后你沿着这棵树往下走，这样就能比较容易地解释它，这使得它在应用环境中非常容易使用，在这种环境中，你不仅要解释结果，还要解释你是如何实现这一结果的。还有人吗？有还有吗？所以相对来说对异常值不敏感不敏感？你知道我的意思吧？对吧对，不敏感感觉就像是对那种感觉麻木不仁 但不是的那我们是什么意思？是的与其他方法相比，它对异常值相对不敏感。话虽如此，但这确实有点取决于所使用的决策规则。因为我们有这种贪婪的方法，你可以多次使用相同的变量，比如说，所有这些都会导致对离群点不敏感，但对非常主要的变量却很敏感，如果这说得通的话。因此，在某种程度上，对异常值也有一种敏感性。因此，如果有一个变量、一个特征在整个空间中占主导地位，那么它就会被反复使用。而你对其他变量的了解并不多，这可能是一个缺点。但你说得对，它对异常点相对不敏感。是啊。所以我又补充了一点，我认为它的缺点是对主导变量不敏感。我的意思是，举例来说，如果一个变量在解释某种关系方面非常出色，那么它就会被多次使用，这也是可以的。这其实没什么。这并不是一个真正的问题。它仍然有很好的预测作用，但你却忽略了可能稍弱但仍然重要的联系，这可能会影响你对结果的解释。还有其他人吗？是的，它既可用于定性，也可用于定量，有时我们不必使用虚拟变量。在数据类型方面，它真的很灵活。因此，它的最大优点就是没有标准化，没有标准化，因为每个变量都是单独研究的，所以在每个拆分步骤中都是独立的。因此，这意味着，即使变量的测量尺度真的不同，也没有关系，因为如果你分别研究它们，而不是进行比较，那么就不会对模型产生影响。所以不需要标准化，这很方便。是的，就像我们在分类类中看到的那样，可以使用序数数据、数字数据来处理所有数据。它会完成它的工作。还有别的吗？还有目光短浅还有这是一种短视。完全正确所以，这基本上就是我刚才说的否定句。所以我们只我们只看好的开始和这个特定的时刻。我们不考虑其他任何事情。这是非常短视的是的很难相信微小的变化你能吗？是的，这经常发生是的 这很有趣 因为一方面 我们可以说它们对异常值相对不敏感但另一方面，你又可以很快得到截然不同的树。所以我认为它们是一种。可能会有很多变化。我们的意思是，如果你有相同的数据，你基本上会得到不同的方法。很多时候，如果你多次尝试算法，因为这取决于在那个特定的时刻，比如说，它在寻找什么样的变量。因此，有时如果你有一个数据集，正在进行交叉验证，然后你在第一部分建立一个模型，并在第一个分割中使用不同的变量，然后在第二个交叉验证集上建立一个模型。这样一来，模型树就会完全不同，因为一切都会受到第一次拆分的影响。因此，以棒球为例，年龄是一个很好的预测变量。如果我们从整个数据集来看的话。但是，如果你再考虑一下，好吧，如果我们做的某种人群恰好有更多来自右侧的训练数据，那么突然之间，年龄就不是一个很好的预测变量了。因此，我们就不会进行影响整棵树的第一次分割。因此，在这种情况下，整棵树看起来就不一样了。这是否意味着模型会过度拟合？是的，树很容易过度拟合。所以我们迄今为止看到的这些树都是单一决策树，一棵树作为模型，一棵树作为决策树，很容易出现过拟合。真的很容易。所以使用单一决策树时一定要小心。我们已经讨论了很多克服这个问题的方法。所以我们谈到了在树叶中寻找最少的观测值。我们还谈到了早期停止，即当基尼不纯度或 RSS 降低得不够多时，我们就停止。我们还谈到了修剪我们的树，因为是的，过度拟合。还有什么要补充的吗？你想继续吗？继续好的我觉得数据丢失了是的，这取决于。这取决于缺失数据在哪里。如果数据缺失在最重要的考虑变量中，那么在这种情况下，数据缺失就会有点困难。但这不是一个根本问题，因为这对模型本身不是问题。所以它不会影响模型的有效性或稳定性。显然，它可能会影响预测能力。因此，还有其他一些方法，比如神经网络，在数据缺失的情况下非常吃力。因此，这就影响了你需要做多少预处理。例如，神经网络往往比决策树需要更多的预处理。决策树就是把你的数据丢进去，希望能得到有用的结果。好了，我们继续。到目前为止，我们已经讨论过决策树了，我们说好吧，决策树挺酷的。我们可以将其可视化。它们很直观。效果也不错。但是实际上，在某些情况下，它们并不是很好的预测工具。所以我们的想法是，如果一棵树并不聪明 那如果我们用更多的树会怎么样呢？当然，这就是解决方案，也是我们所有戏剧和问题的答案。事实证明，是的，这就是解决办法，尽管听起来有点愚蠢。因此，我们现在将讨论结合多个弱学习器的一般方法，以全面提高模型的预测能力。因此，我们也可以将这些学习者组合称为合奏，就像音乐家们聚在一起合奏一样。因此，我们也称这种方法为合奏学习。因此，我们使用的不是单一模型，而是模型群。其中有两种方法，即袋式学习和随机森林。因此，这背后的很多想法其实都可以追溯到我们在第四讲中谈到的内容。在第四讲中，我们谈到了重采样和自举的概念，以及我们如何创建多个训练数据集，以真正稳健的方式训练我们的模型。你还记得自举的概念吗？我们从可能的数据池中反复抽样，创建训练数据集，并进行替换。所以，你还记得当你学习统计学和概率论的时候，你有一些带颜色的球，你把它们取出来，然后记录下来，再放回去，然后再取出一个。记录就是引导，速度非常快。你把记录拿出来，放到训练数据集中，然后再放回去，然后再拿出来，再放回去。因此，在训练数据集中，多条记录可能是相同的。它可以出现多次。这就是交叉验证的最大区别。所以，你是 K 折交叉验证。我们对训练数据进行拆分。这些是我们保留的分割数据。然后对其中的一部分进行训练。你把它放回去，再对其中一部分进行训练。因为你总是会替换这些样本。你可以对相同的记录进行多次训练。bootstrapping 的理念也可以用于模型训练，如果我们这样做，我们就称之为 bagging。所以，bagging 实际上是 bootstrap aggregation 的缩写。我一直认为，它的意思是，我们把一个回溯数据加入多个模型中，然后再混合一下。我就是这么记的。但实际上，它代表的是引导聚合。所以，尽管我们是在决策树的语境下谈论 "bagging "的，但它也可以用于不同的模型和不同的学习过程，因为它只是一个概念，是我们构建模型或使用模型的一种通用方法。所以我们称它为集合模型，因为它结合了多个弱学习者，可以应用于不同的模型，而不仅仅是决策树。所以，bagging 背后的理念是，我们希望引入随机性元素。随机性实际上是你的朋友，因为随机性可以提高模型在数据上的可靠性。例如，在这种情况下，我们说我们有多个观测值和独立观测值。这就是你的数据。你有某种方差西格玛。换句话说，如果我们对一组观测数据进行平均，就可以减小方差。这背后的想法基本上是，如果你回想一下我们正在寻找的数据分析原理，我们如何预测一个分布的平均值。举例来说，如果你收集的数据越来越多，比如你有很多重叠的分布，那么在某些时候，你就能很好地预测人口的真实平均值是多少。如果你反复采样，并查看所有这些不同分布的平均值，你就能得到一个很好的预测结果。因此，这基本上就是我们的想法，但我们并没有这些不同的样本和多个不同的训练数据集，可以用来预测例如校长讲座中分布的平均值。因此，我们的想法是使用 bootstrapping 方法，也就是我们之前说过的自举法。我们从训练数据集中重复抽取样本，并进行替换。这样，我们就能创建多个引导训练数据集。然后，我们在每个数据集上分别训练我们的方法。然后求出所有预测结果的平均值。这就是我前面提到的例子中对平均值的整体预测。但实际上，我们要做的是预测一个特定的值，例如决策树。因此，我们的想法是，如果在数据的多个子集上建立多个模型，就能建立一个更稳健的模型。因为你基本上是在减少其中的方差，而这种将多个模型组合在一起，然后平均它们的预测结果的想法，就是我们所说的 "套袋"。这就是我们可以用于不同模型的方法。因为你基本上可以组合任何结构的多个学习器。不一定非得是树状结构。因此，此时的预测值可以是一个数字预测值。但你也可以将其用于分类树。因此，它也可以是一个类标签。然后，这一点的类标签将由多数票决定。因此，所有这些不同的方法、不同的模型基本上都是求平均值。要么是预测值的平均值，要么是最频繁的投票。是的所以这三个集合，你真正要做的是决定要组合多少棵树，使用多少个引导样本，然后在这些样本上训练你的树，然后组合所有的平均值。这就是公式中的参数 b。那么，我们要合并多少个样本呢？简单地说，实际使用的次数越多，就越准确。尽管这有一点不平衡。所以在某些时候，你并不会得到更多的改进。但显然，计算时间越长，情况就越相同。原理是一样的，树越多，样本越多，精度越高，计算时间越长。因此，你必须根据数据的复杂程度来选择树的数量，变化越多，数据越复杂，树就越多，问题就越简单。必要的树数量越少越好。所以，我之前提到的一个非常有趣的观点，你可能已经注意到了，但也可能没有。正如我所说，我们结合了弱学习者。我指的是树本身。它们其实并不好。例如，在这种情况下，我们把它们种得很深。我们没有对它们进行修剪。它们有很多过度拟合的树，但我们有很多不同的过度拟合跟踪。因此，所有的树都是在数据的子集上独立训练出来的。这就意味着一棵树的拟合效果非常糟糕，远远超过了数据的拟合效果。但如果你把所有的树组合起来，然后取平均值，它们就会做得很好。因此，这就是 "套袋 "技术的原理，它的基本定义是将大量糟糕的模型进行套袋，但通过对它们进行平均，就能得到相当不错的预测结果。换句话说，每一棵树都有较高的方差和较低的过拟合偏差。但我们说，通过将它们组合在一起，可以降低模型的方差。是的。所以这就相对简单了。显然，这是对一个袋装模型的检验。我们的想法是将这些树反复拟合到数据的自举子集上。因此我们可以证明，平均而言，每个工厂都使用了大约三分之二的观测数据。这里面有一个数学证明，我不打算给你看，因为很无聊，但它确实存在。因此，每棵树剩下的三分之一观测数据基本上都没有被使用。这就是所谓的 "缺失观测"。这也是我为什么把 "回溯 "当作 "袋装 "的另一个原因，我不知道，这真的很傻。这张幻灯片的意思是，每棵树都有不同的测试数据集。基本上，每棵树都使用了不同数据的不同部分，不同数据的不同子集。你必须确保你对这棵树进行了实际测试和预测。然后，在三次预测中得出 B 左右的结果，并求取平均值。所以，这告诉你的就是，一定要使用测试数据来衡量准确率。然后我们对多个模型进行平均。好吧，如果我们有很多非常糟糕的过度拟合树，而它们在一起可以实现一些伟大的目标。当我第一次听到然后读到随机森林的时候 我想到了一件事 你们都看过 "魔戒之王 "吧？这些树聚集在一起 然后共同做出决定而每一棵树，我的意思是，在很多事情上，它们都很聪明，但它们并不优秀，不是吗？因此，我们希望通过在决策过程中将所有这些树结合起来，从而实现良好的预测效果。这就是随机森林的理念。所以，如果你听到随机森林，你就会想到很多环的末端。这是非常准确的。随机森林的理念建立在回溯的基础上。所以，如果你了解回溯，你就会了解随机森林。我们现在做的主要工作是尝试将踪迹关联起来。所以我们说随机性是好的。所以我们不希望我们的模型之间存在某种联系和关联。我们希望模型中的随机性越大越好，因为它能提高我们的预测能力。所以和以前一样。我们仍然在自举训练样本上建立许多树。但这次，我们不再以同样的方式构建这些树。我们把数据给他们，然后告诉他们，你们可以进行分割，使用基尼法或其他方法。现在，我们实际上告诉他们，好吧，你来做分割，但你只能使用我给你的变量子集。你不能使用所有变量，只能使用其中的随机子集。这样做的结果就是迫使这些树改变它们的行为。突然间，每棵树的构建方式都不一样了，因为它们必须使用不同的变量。因此，这实际上只是增加了预测树可以使用的预测因子的随机性，为你的模型引入了更多的随机性，这在很多情况下提高了你的预测能力。是的，每一步都会首先选择预测因子。因此，在建立模型的过程中，每一步都会引入越来越多的随机性。你还记得我们之前说过，年龄是一个很好的分化变量。因此，这些树中的每一棵都可能会根据年龄进行分割。但如果我们告诉他们，好吧，你们不能使用年龄作为变量，那会发生什么呢？你们必须使用不同的变量子集。这样一来，突然间所有这些树都彼此不同了，它们都会告诉你一些关于数据空间的略微不同的信息。这就是随机森林背后的理念。因此，后树、多树的平均性能相当不错。随机森林，是指在每个点上随机分裂出多个随机树，然后平均它们的性能。每一次分割都会增加随机性。每一步都是随机的。这使得随机森林在很多情况下成为最强大的预测器和最强大的分类器之一。我这么说是谨慎的，因为这并不意味着随机森林总是你能使用的最佳模型。这意味着，在许多文献中，当你看到社会科学模型之间的比较时，随机森林的表现要优于其他方法。我的意思是，它们的表现就像回归和 K 等所有方法一样。这并不意味着随机森林总是最佳模型。我们已经说过，最佳模型取决于你的数据和背景，以及你想要实现的目标。但我总是说，如果你的数据非常复杂，而你至少没有尝试过随机森林，那你就有点，我认为你会错过一些东西。所以，关于我的发票，我还想说的一件事，我没有放在这里，那就是：随机森林和乞讨。随机森林和乞讨。所有这些方法都有一个主要缺点，那就是可解释性。我们已经说过，我们有所有这些树，我们把它们放在一起，然后对它们进行投票。但是，你失去了原有决策树的所有可解释性。所以，这就是一种漂亮的树形结构。你可以清楚地看到每一个点的去向。所以，你有了所有这些树，成百上千棵。这不可能可视化，也极难解释。所以这也是另一点，我们说这是一个很好的预测指标，但它们并不擅长解释。因此，有一些方法可以稍微克服这一点。比如，你可以看一下特征的重要性，它是衡量一个变量对模型预测能力的重要性。但即使是这样，也比不上一棵完美的树，它能准确地告诉你它在做什么。所以是的，这取决于随机森林是否适合你。这也完全取决于你是否重视可解释性。现在，让我们用最后五分钟来简单谈谈提升。提升法也是一种通用方法，所以就像bagging一样，它不仅可以应用于竞赛，也可以应用于其他方法。你应该还记得，bagging 涉及到多棵树，然后使用 bootstrapping 对原始训练数据进行多份拷贝，分别拟合，将所有树组合成一个袋子，摇一摇，看看平均准确率。因此，这些树中的每一棵都是在彼此独立的数据基础上建立的。我们说，这可能会导致这些树看起来非常相似，并且存在某种相关性，这就是为什么森林和提升的发展实际上将这一想法带入了不同的方向。所以它说，好吧，但如果我们不分别建立它们，而是按顺序建立呢？如果我们用一棵决策树来做决策，它做得还算不错，但却遗漏了我们空间中的一些信息。但如果我们在此基础上再建一棵树，它就能覆盖这棵树表现不佳的遗漏空间，这就是提升背后的理念。所以，让我们来看看这个，因为这更有趣。提升法背后的直觉是，我们不能只拟合一棵树或一组树，而是要有顺序地、一步一步地慢慢改进我们的学习。因此，在给出一个当前模型时，我们首先拟合一个残差。我们首先拟合一棵决策树。这就是我们的普通模型方法。然后我们再对残差拟合决策树。这就是原始树的误差。然后将其添加到拟合函数中。所以我们的想法是，每一棵树都非常小。因此，它更像是一个小部分的树叶和节点集群，只要有遗漏的空间，它就会被添加到现有的树上。因此，我们对残差进行了拟合，因为残差就是我们的预测误差。这就是我们的遗漏。这就是我们在模型中没有捕捉到的。我们通过拟合这两棵小树，来准确改善我们遗漏的地方。所以，如果我回想一下，比如说，从一开始我们的例子，我们有这样一个例子，我们说我们可以建立一棵树，把它分成这三个空间。但你可以看到，我们实际上错过了，比如上面这些点和下面这些点。因此，我们的想法是提升这一点。我们说这些点对我们很重要。而模型中存在一个错误。因此，我们在迷你树的基础上进一步细分这些小空间。提升法背后的理念是，如果我们重视数据中的这些小复杂性，如果你有很多你重视的复杂性，你想在你的模型中捕捉到这些复杂性，而你的模型还没有捕捉到，那么你就可以在现有模型的基础上使用提升法来进一步改进它。对于那些遗漏的小型怪异数据空间。好了。差不多完成了。总结。我们已经了解了决策树的优点。你可以对其进行解释和可视化。它们真的很酷，因为它们可以处理很多数据。我们还了解到，决策树有时可能有点模糊。因此，在大多数情况下（不是全部，是大多数情况），组合树能极大地提高它们的性能。显然，例子中考虑的树的数量应该与你的问题和数据复杂度相适应。复杂度越高，树就越多。这样做的缺点是缺乏可解释性。这也是风和森林以及一般袋装方法的主要缺点，因为我们无法解释单棵树，而必须依赖于特征重要性得分等。等等。因此，在计算机实验室中，我们将研究这些决策树、单一决策树和组合方法的实施。这也将让我们有机会讨论这些方法和参数，并对其进行测试。那么，我们应该使用多少棵树呢？决策树应该建多深？等等。等等。所有这些都将在计算机实验室中讨论。下周我们将讨论支持向量机。好的，祝大家一周愉快，明天见。