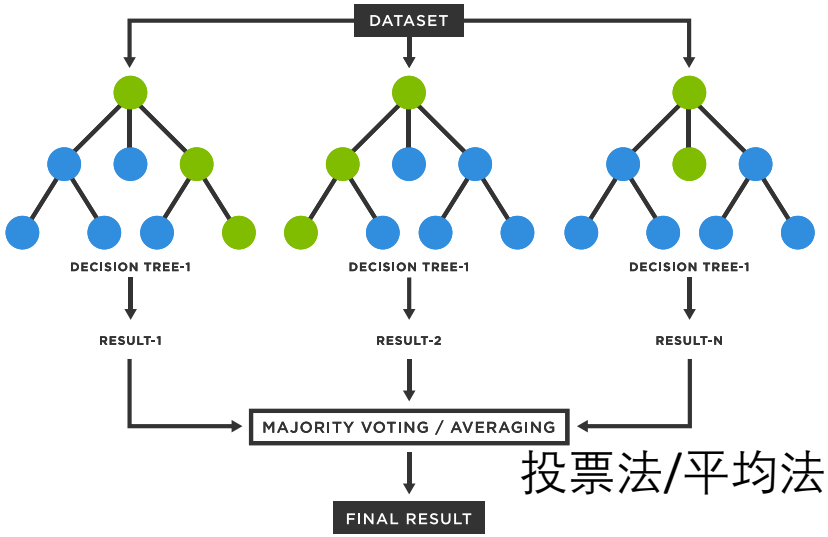
随机森林

随机森林

- 随机森林（Random Forests, RF）由多棵决策树构成，每棵树都对数据进行独立的**分类**或**回归**预测。
- 随机森林计算开销小，但它在很多任务中展现出强大的性能，被誉为“**集成学习**技术的代表方法”。

- 森林?



- 单个决策树容易因数据的波动而表现不稳定，而随机森林通过集成多个决策树，利用平均效应减少了模型的总体方差，从而使预测更为稳定可靠。

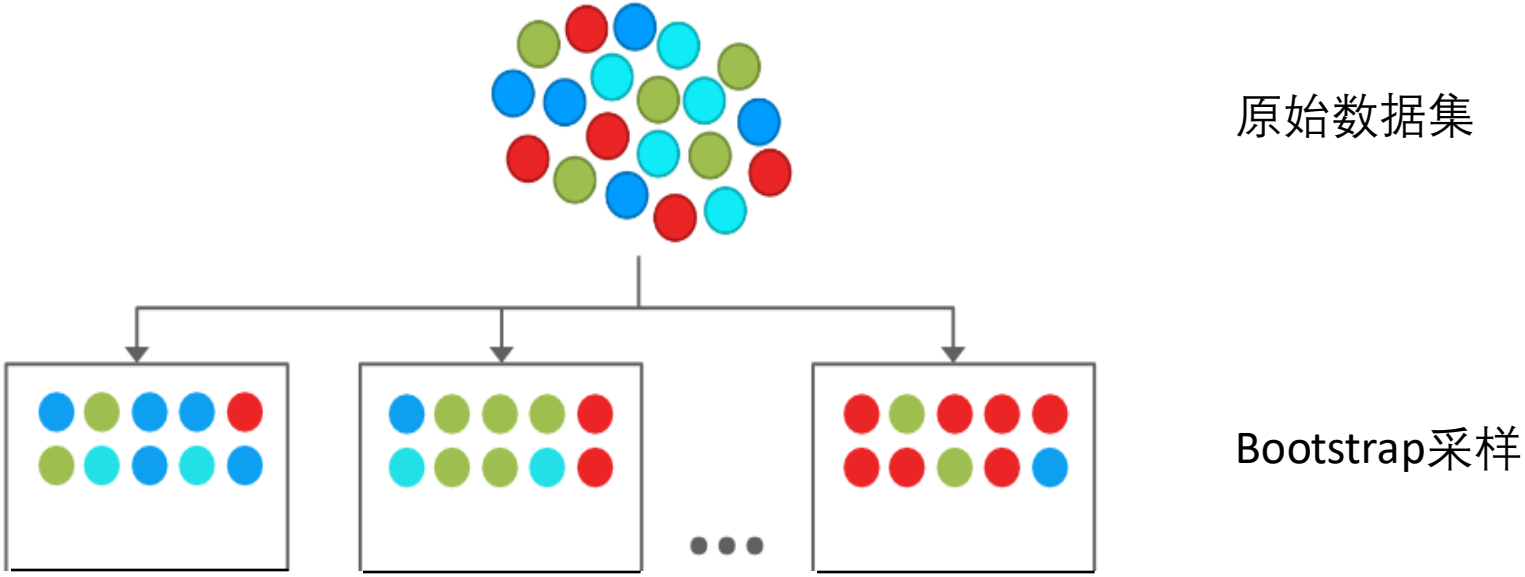
随机森林的“随机”

- 在随机森林算法中，**需要注意单个决策树之间的相关性问题**。如果多个决策树在同一个数据集上训练，且每一步都选择最佳分裂点，那么它们可能会变得**强相关**!
- **为了提高集成模型的泛化能力，每棵树应尽可能地独立**。随机森林通过以下方式引入随机性以确保决策树的独立性：

- **数据**的随机性选取
- **树节点特征**的随机选取

数据的随机选取

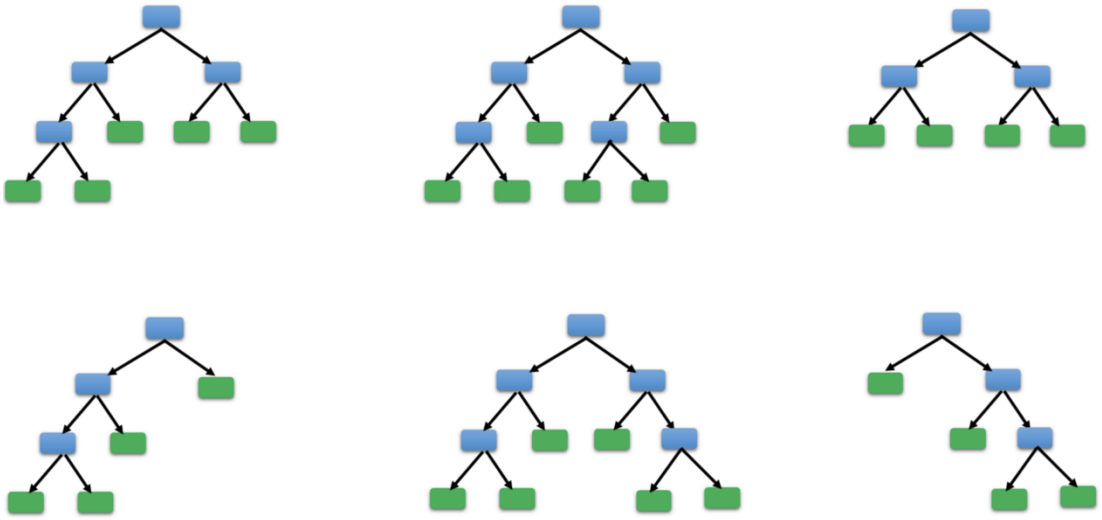
- **Bootstrap采样（Boostrap sampling）**是一种从原始数据集中进行有放回抽取的样本抽取方法，主要用于估计一个统计量的分布。
- 随机森林使用**Bootstrap采样来生成多个不同的样本子集**：从包含*N*个样本的原始数据集中随机抽取一个样本，并在记录该样本后将其**放回数据集**，这个过程重复*n*次，即可以获得包含*n*个样本的集合（在随机森林中，$n = N$）。



数据的随机选取

- 为什么要用Bootstrap抽样?

1. **降低决策树间的相关性**：自助采样法通过为每棵树生成不同的训练集，保证了每棵树都有不同的视角来看待数据，减少了树与树之间的相关性，从而提高了泛化能力。



数据的随机选取

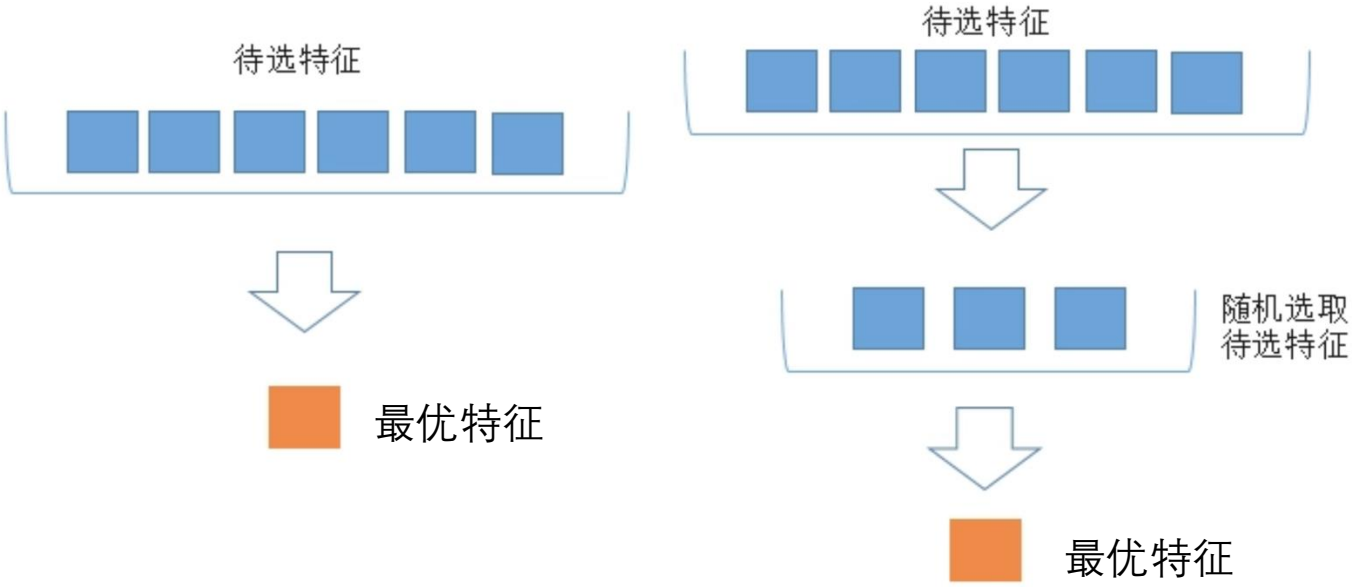
- 为什么要用Bootstrap抽样?

2. **包外估计**：在自助采样法中，那些从未抽到的样本大约是原始数据集的36.8%，可以用来估计模型的泛化误差，而不需要准备额外的测试集。我们把这部分数据称为“包外”（Out-of-Bag, OOB）数据。

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}=\frac{1}{e}\approx 0.3679$$

节点特征的随机

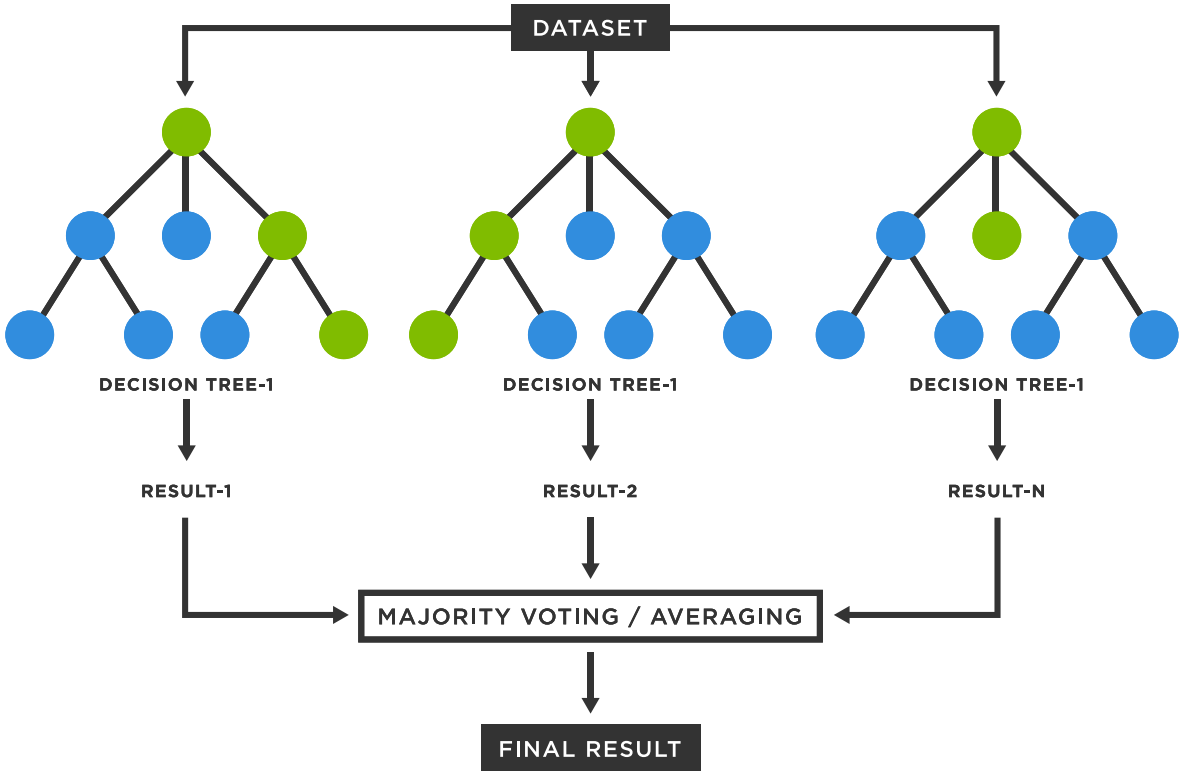
- **随机选择属性子集**：在随机森林中，每个决策树在选择节点特征时首先从待选特征中随机选取 k 个特征作为特征子集，然后从这个子集中选择最优节点特征。
- **选择 *k* 值的推荐**：一般$k = log_2 d$或$\sqrt{d}$，其中 d 是属性的总数（经验法则）



这一机制迫使随机森林模型探索数据中不同的特征组合和不同的树结构。

随机森林的特点分析

- 随机森林通过**在多个决策树上分布计算任务**，能够提高处理大型数据集的效率。（随机森林具有天然的并行性，可以在多个处理器或分布式环境中同时进行，从而显著提高训练速度。）

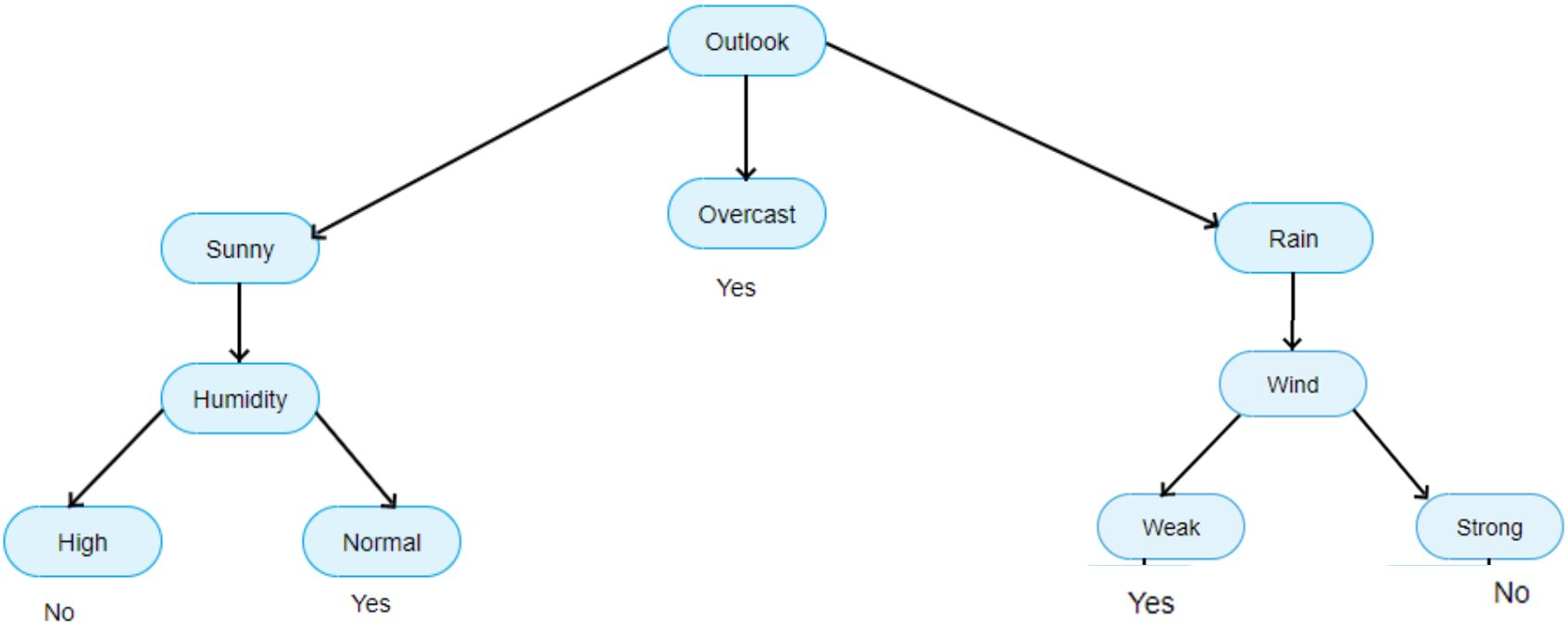


并行的构建树以及并行的推断（分类/回归）

随机森林的特点分析

能评估并确定各特征对分类的重要性。

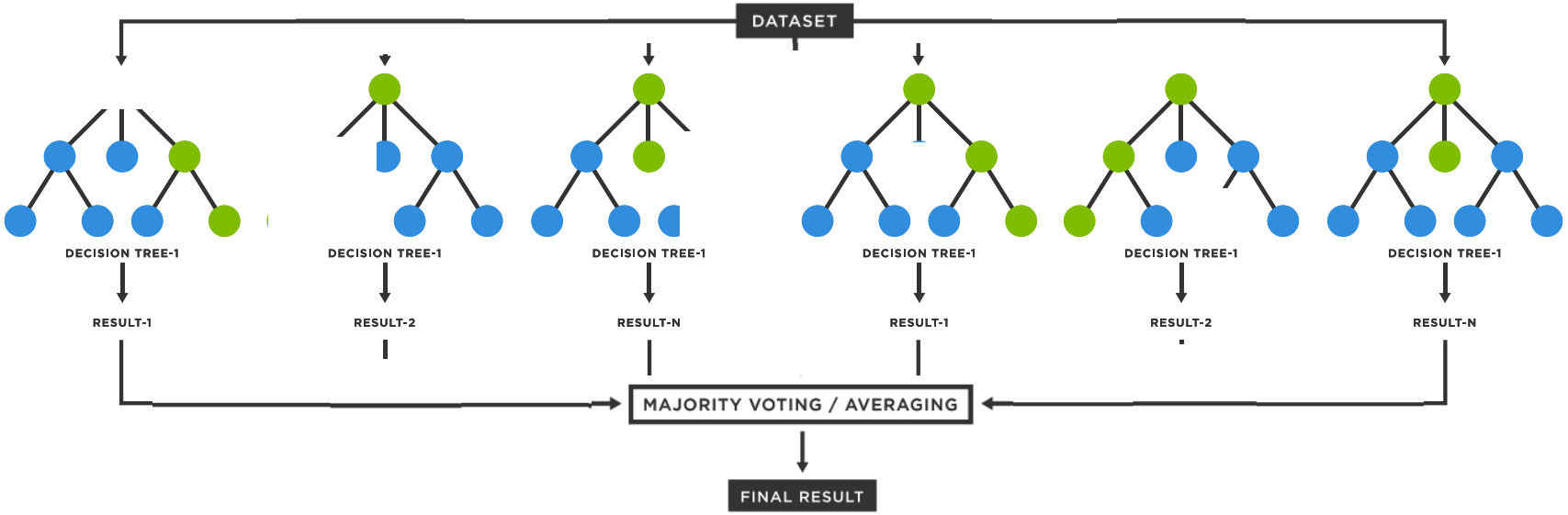
- 在每棵决策树中，当一个特征被用来分裂节点时，会使该节点的纯度提高（例如**降低基尼指数**或**信息熵**）。随机森林统计每个特征在所有树中引起的不纯度降低的总量或平均值，这个量就代表了该特征在整个模型中的贡献和重要性。



随机森林的特点分析

数据部分缺失的情况下也能保持一定的准确性

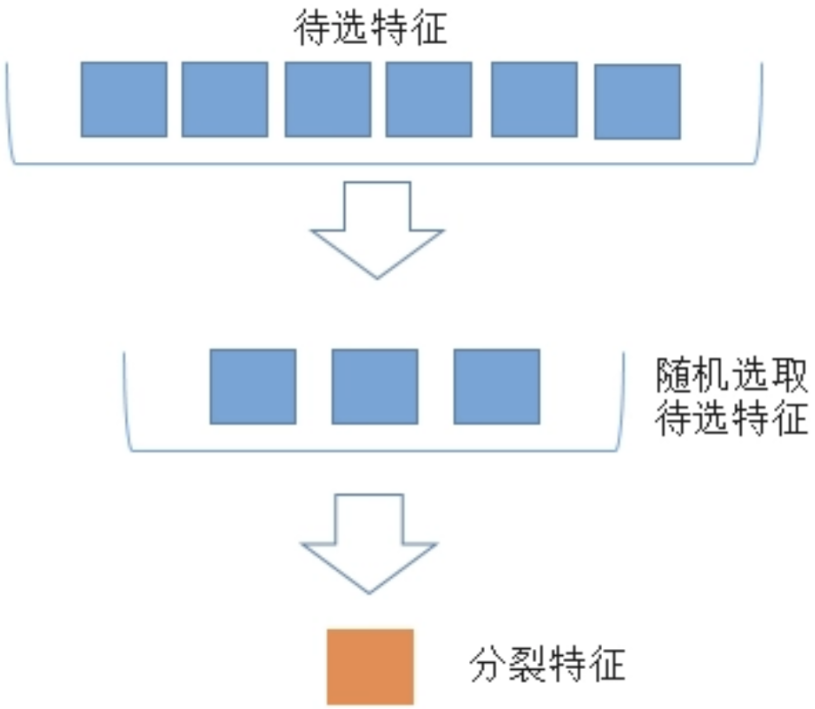
- 每棵树的训练数据是通过bootstrap抽样获得的，即使部分样本在某些特征上缺失，也不会同时影响所有树。
- 在每次节点分裂时，随机森林只考虑随机抽取的部分特征。如果某个特征存在缺失值，而这棵树恰好没有选取该特征作为分裂依据，缺失问题对这棵树就不会有影响。



随机森林的特点分析

随机森林在处理高维数据表现出色。

- 在每个节点进行分裂时，随机森林不会考虑所有的特征，而是从全部特征中随机选取一个子集进行比较。这种做法降低了单次分裂的计算复杂度。在特征数量很多的情况下，也能减少高维数据中冗余和噪声特征的影响。



课堂练习

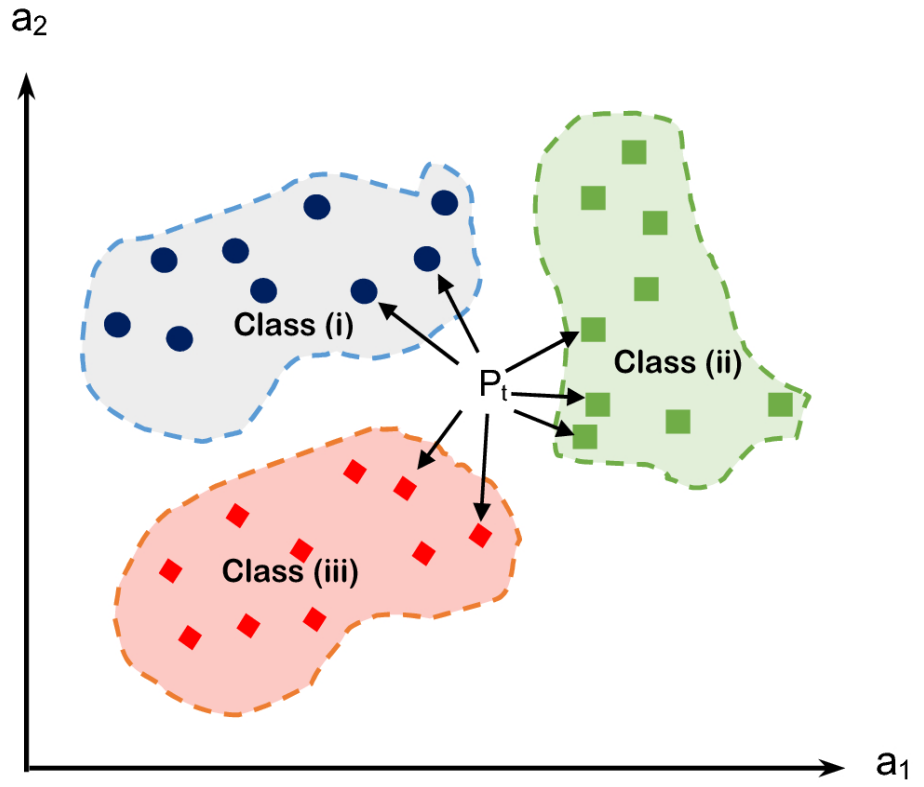
判断以下问题的对/错

1. 随机森林通过构建一个决策树来减少模型的方差。
2. 在随机森林的构建过程中，每棵树的训练数据是从原始数据集中通过有放回抽样（bootstrap sampling）得到的。
3. 随机森林模型中的所有决策树在节点分裂时会考虑所有可用的特征。
4. 随机森林模型通常比单一决策树模型具有更高的计算成本，因为它需要训练多棵树。
5. 随机森林能够减小单一决策树模型方差，但不能减小偏差。

K-近邻算法

K-近邻算法（k-nearest neighbors）

- K近邻算法（k-NN，k-Nearest Neighbors）的核心思想是非常直观的：通过测量不同特征值之间的距离来进行分类或者回归。



问题：点P_t属于哪个类别?

$k = 7 \quad \Rightarrow \quad P_t \in$ 类别(ii)

K-近邻算法

当需要对一个新的样本进行分类时，算法会在特征空间中按照某种距离度量找到最接近这个新样本的k个已知样本（即“邻居”）。在分类任务中，**k-近邻算法通常采用“投票法”；**对于回归任务，**k -近邻算法则使用“平均法”**。

特点：

- **简单，易实现**
- **无参数**
- **懒惰学习**

三要素：

- **距离度量**:
- ***k*值的选择**:
- **决策规则**。

距离度量

- 在数据科学中，特征空间中两个实例点之间的距离反映了这两个实例点的相似程度。距离越近，表明这两个实例在特征空间中更相似。

- 距离的度量有多种方式，如**欧几里得距离**、**曼哈顿距离**、**余弦相似度**等。K近邻模型使用的距离一般是欧氏距离，但也可以是其他距离，如更一般的$\boldsymbol{L_p}$**距离**或**Minkowski距离**。

R^n空间的两个点x_i,x_j的$\boldsymbol{L_p}$**距离的定义**

$$L_p\left(x_i, x_j\right)=\left(\sum_{l=1}^{n}\left|x_i^{(l)}-x_j^{(l)}\right|^p\right)^{\frac{1}{p}}$$

这里的$p \geq 1$

当$p=1$，称为曼哈顿距离

$$L_1\left(x_i, x_j\right)=\sum_{l=1}^{n}\left|x_i^{(l)}-x_j^{(l)}\right|$$

距离度量

当$p=1$，称为曼哈顿距离

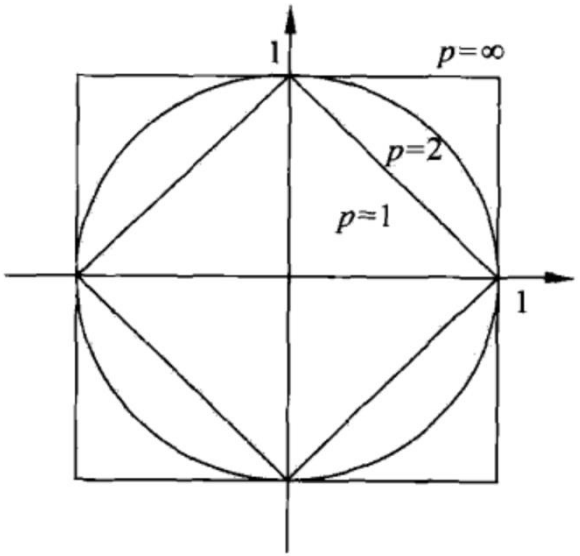
$$L_1(x_i, x_j) = \sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|$$

当$p=2$，称为欧式距离

$$L_2(x_i, x_j) = \left(\sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|^2 \right)^{\frac{1}{2}}$$

当$p=\infty$，称为切比雪夫距离

$$L_\infty(x_i, x_j) = \max_l \left| x_i^{(l)} - x_j^{(l)} \right|$$



左图给出了在二维空间p取不同值时，与原点的L_p距离为1的点图形。

距离度量

$$L_p\left(x_i, x_j\right)=\left(\sum_{l=1}^{n}\left|x_i^{(l)}-x_j^{(l)}\right|^p\right)^{\frac{1}{p}}$$

已知二维空间的3个点$x_1=(1,1), x_2=(5,1), x_3=(4,4)$，试求在$p$取不同值时，$L_p$距离下$x_1$的最近邻点。

距离度量

$$L_p(x_i, x_j) = \left(\sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|^p \right)^{\frac{1}{p}}$$

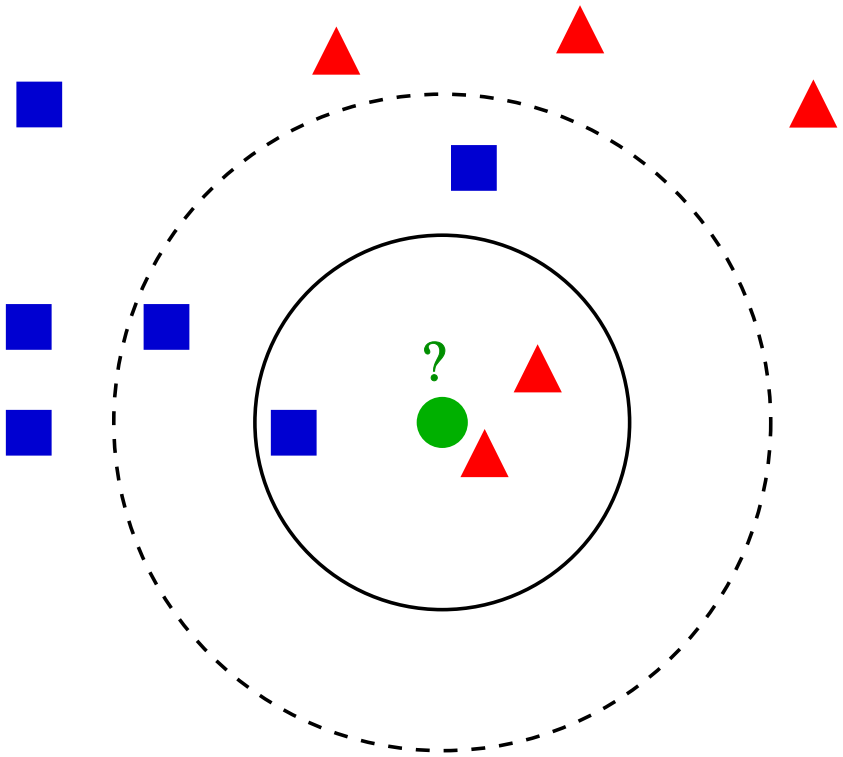
已知二维空间的3个点$x_1 = (1,1), x_2 = (5,1), x_3 = (4,4)$，试求在$p$取不同值时，$L_p$距离下$x_1$的最近邻点。

当$p = 1$时	当$p = 2$时	当$p = 3$时	当$p = 4$时
$L_1(x_1, x_2) = 4$	$L_2(x_1, x_2) = 4$	$L_3(x_1, x_2) = 4$	$L_4(x_1, x_2) = 4$
$L_1(x_1, x_3) = 6$	$L_2(x_1, x_3) = 4.24$	$L_3(x_1, x_3) = 3.78$	$L_4(x_1, x_3) = 3.57$
L_1距离下x_1的最近邻点x_2	L_2距离下x_1的最近邻点x_2	L_3距离下x_1的最近邻点x_3	L_4距离下x_1的最近邻点x_3

***由不同的距离度量所确定的最近邻点是不同的**

K值选择

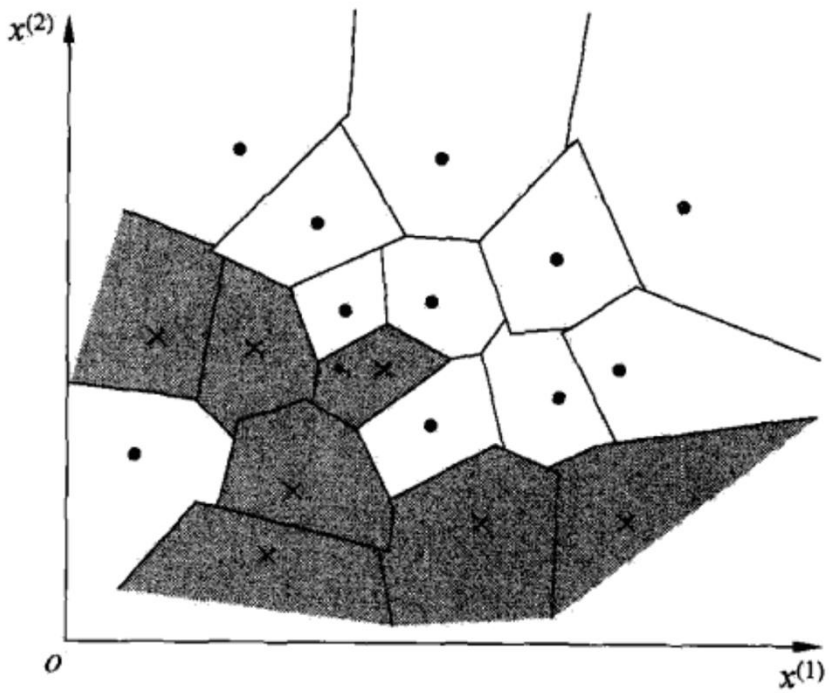
- 不同K值会对结果造成影响



*k*近邻算法例子。如果k=3？如果k=5？

K值选择

- 当**训练集**，**距离度量**，**k值**，**决策规则**确定的情况下，K近邻算法相当于给特征空间分割成了若干个子空间，新的样本落到了哪个子空间中，就属于这个字空间的类别（分类问题）。



当k=1（最近邻）时，二维特征空间被分割的例子

K值选择

- **较小的 *k* 值?** 特征空间被分割成非常多的子空间，模型复杂。容易发生过拟合。
- **较大的 *k* 值?** 特征空间被分割成若干（少数）的子空间，每个子空间非常大，模型非常简单。容易欠拟合数据。

结论：K值即不能过大也不能过小

- **经验法则**：选择样本点总数的平方根作为 k 值。例如，如果有 100 个样本点，可以尝试使用 k=10。
- **二元分类：**选择奇数的 k 值可以避免平票的情况，确保总是有一个类别占据多数。

决策规则

在找到K个邻居后如何做决策?

一般情况：投票法/平均法

处理平票情况:

1. 方法1：选择这些类别中距离最近的邻居的类别。
2. 方法2：随机选择一个类别或考虑其他k值。

距离权重:

1. 在某些k-NN算法变体中（Weighted K-NN），在决策时可以给予更接近的邻居更大的权重。例如，可以使用邻居距离的倒数作为权重。

K-近邻算法的缺点

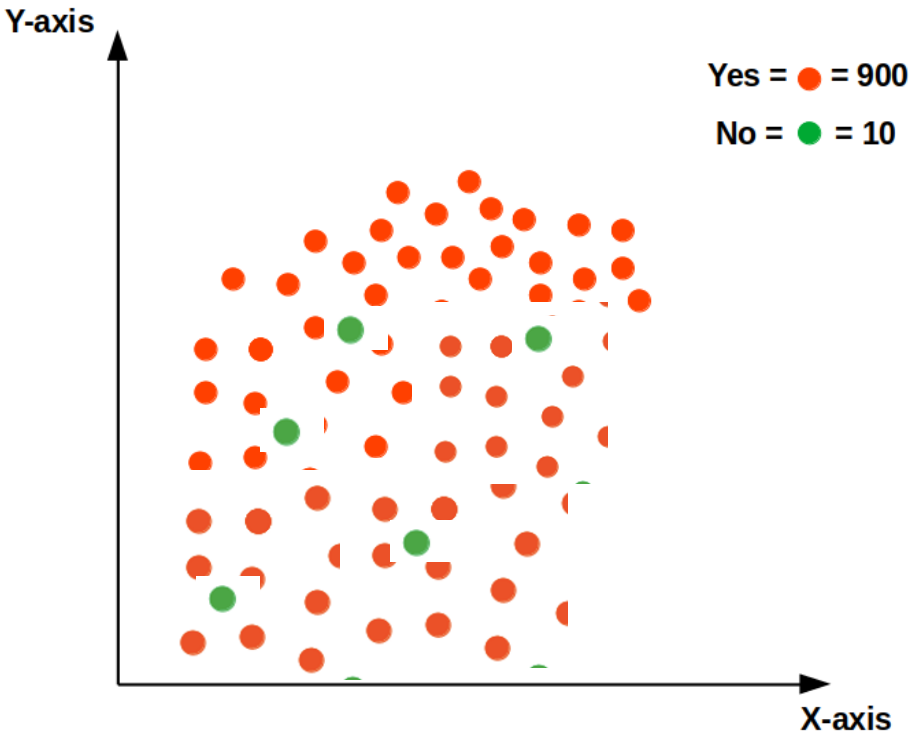
- **K-NN算法的计算效率低**:

1. **遍历整个训练集：**
   对于每个新样本，算法需要计算它与训练集中所有样本之间的距离，当训练数据量很大时，计算量会非常庞大。
2. **高维数据问题：**
   在高维空间中，距离计算不仅代价高，而且常常受到“维度灾难”影响，使得距离度量失去区分性，进一步增加了计算负担。
3. **缺乏模型简化：**
   与其他需要预训练模型的算法不同，K-NN没有显式的模型构建过程，所有工作都在预测阶段完成，这意味着无法通过提前计算或模型压缩来降低预测时的计算复杂度。

K-近邻算法的缺点

- **不适用于不平衡的数据集**

在不平衡的数据集中，少数类样本周围很可能被多数类样本包围，使得少数类样本的类别信息被“淹没”，而多数类样本占主导地位，所以KNN会更倾向于预测为多数类。



K-近邻算法的缺点

- **特征比例影响**:

k-NN是基于距离的算法，如果一个特征的范围比其他特征大得多，那么它可能会对距离计算产生不成比例的影响，从而影响算法的性能。因此，在应用k-NN之前一般要进行特征缩放（如标准化或归一化）。

举个例子，假设有一个信用评级数据集，其中包含两个特征:

- **年龄**：范围大约在20到70岁之间
- **年收入**：范围可能在50,000人民币到500,000人民币之间

在这种情况下，如果直接使用k-NN算法计算欧氏距离，由于年收入的取值范围远大于年龄，其差异在距离计算中会起到主导作用。这意味着，即使两个样本在年龄上差异明显，但只要它们的年收入接近，它们之间的距离也可能很小，从而影响到k-NN的分类效果。