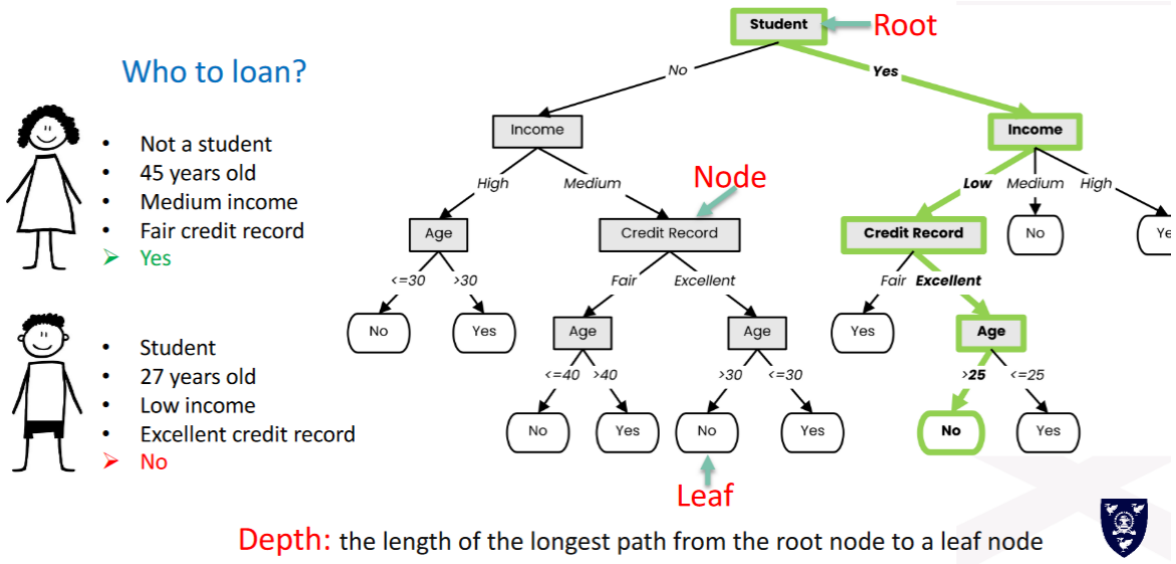
INT104W8_决策树与集成学习v0.2

决策树

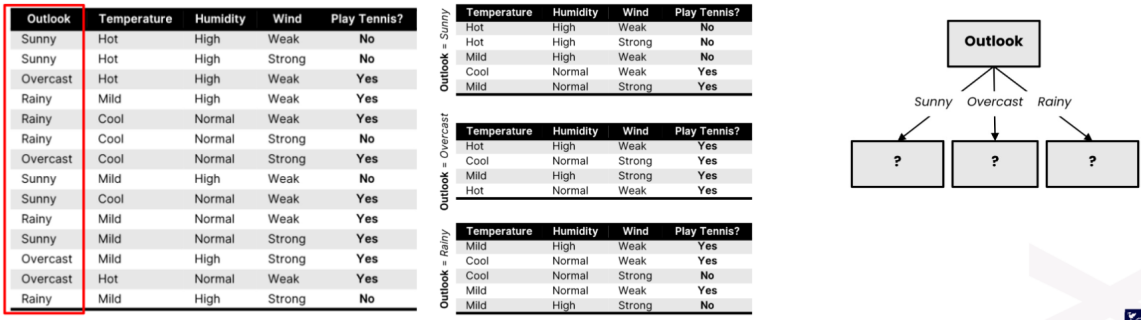
决策树是一个树状模型，用于说明导致某些决策的一系列事件。

每个节点（Node）表示对属性的测试，每个分支都对应该测试的结果。



决策树学习（Decision Tree Learning）

通过对有标签数据的监督学习来获得适合未来决策（泛用性）的决策树



Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rainy	Mild	High	Weak	Yes
Rainy	Cool	Normal	Weak	Yes
Rainy	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rainy	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rainy	Mild	High	Strong	No

Outlook = Sunny

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

Outlook = Overcast

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	Yes
Cool	Normal	Strong	Yes
Mild	High	Strong	Yes
Hot	Normal	Weak	Yes

Outlook = Rainy

Temperature	Humidity	Wind	Play Tennis?
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No

- **基本步骤**：选择一个属性，并根据其值将数据拆分为更小的集合，递归重复此步骤，直到我们确定标签。

  如在上图示例中，从'Outlook'属性开始作为第一个节点产生三个分支('Sunny', 'Overcast', 'Rainy')，再对各分支测试'Temperature'节点并如此递归或用能直接确定标签的其他特征测试，直到最终能确定标签。

- **选择更好的属性**：选择能提供更好分离的属性。因为该属性分类下得到的子集更纯，知道这个属性的值可以让我们更多地确定标签。（即**得到的子集的熵较低**）

> 想象我们有一个任务，要求判断某种动物是否是鸟类，有特征集{'动物毛色', '是否会飞'}，用常识可以判断显然后者能提供更好的分离。假设测试集中会飞与不会飞的动物比例近似，以动物毛色判断得出的分类结果很可能是：灰色{50%是鸟类，50%不是鸟类}，白色{50%是鸟类，50%不是鸟类}；而以是否会飞判断可以得到：会飞{95%是鸟类，5%不是鸟类}，不会飞{5%是鸟类，95%不是鸟类}。即以'是否会飞'分类得出的子集熵更低。

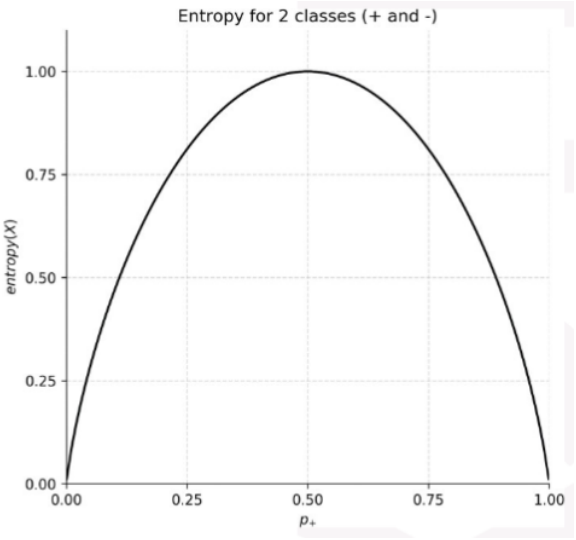
信息增益（Information Gain）

熵（Entropy）

> 在热力学中，熵是体系混乱程度的度量，即系统无序程度的度量。

- 在机器学习中，熵（Entropy）代表了数据的随机性程度（对不确定性的测量）。

  > 如{1,2,3,4,2,3,1,4}的熵就比{1,1,1,1,1,1,1,4}的熵更高。



$$对有k种类别的样本X：\ Entropy(X) = -\sum_{i=1}^{k} p_i log_2(p_i)$$

其中p_i代表了i类元素占总样本的比例。

如果某类元素占总样本的比例非常高或非常低，其贡献的熵就小（如上图）。熵越低意味着可预测性越高（greater predictability）。

信息增益（Information Gain）

在上文决策树学习中，我们曾提到要选择能提供更好分离的属性，因为该属性分类下得到的子集更纯，即子集的熵更低。那么如何衡量每个属性的分离能力呢？我们引入信息增益（Information Gain），属性a（attribute a）的**信息增益代表着**用属性a拆分原数据集后导致**熵减的多少**。信息增益越大，说明这个属性提供的分离性能更好；信息增益越小，说明这个属性提供分离的性能更差。

$$\text{样本}X\text{中属性}a\text{的信息增益：}gain(X,a)=entropy(X)-\sum_{v\in a}^{foreach\ v}\frac{|X_v|}{|X|}entropy(X_v) \tag{1}$$

$$\text{即：信息增益} = \text{信息熵} - \text{条件熵（分类后剩下的信息熵）} \tag{2}$$

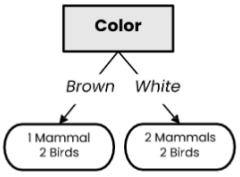
信息增益越高，表示该特征的信息量更大。信息增益量化了选择特定特征使分类不确定性下降的程度。

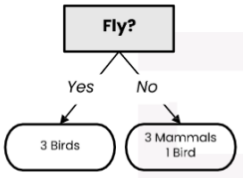
最佳属性 = 具有最高信息增益的属性

计算信息增益练习：

计算并对比'Color'和'Fly'的信息增益：

Does it fly?	Color	Class
No	Brown	Mammal
No	White	Mammal
Yes	Brown	Bird
Yes	White	Bird
No	White	Mammal
No	Brown	Bird
Yes	White	Bird





$$entropy(X)=-p_{mammal}log_2(p_{mammal})-p_{bird}log_2(p_{bird}) \tag{3}$$

$$=-\frac{3}{7}log_2(\frac{3}{7})-\frac{4}{7}log_2(\frac{4}{7}) \tag{4}$$

$$\approx 0.985 \tag{5}$$

$$gain(X,Color)=entropy(X)-\frac{|X_{Brown}|}{|X|}entropy(X_{Brown})-\frac{|X_{White}|}{|X|}entropy(X_{White}) \tag{6}$$

$$\approx 0.985-(\frac{3}{7}(-\frac{1}{3}log_2(\frac{1}{3})-\frac{2}{3}log_2(\frac{2}{3}))+\frac{4}{7}(-\frac{2}{4}log_2(\frac{2}{4})-\frac{2}{4}log_2(\frac{2}{4}))) \tag{7}$$

$$\approx 0.985-(\frac{3}{7}\times 0.918+\frac{4}{7}\times 1) \tag{8}$$

$$\approx 0.02 \tag{9}$$

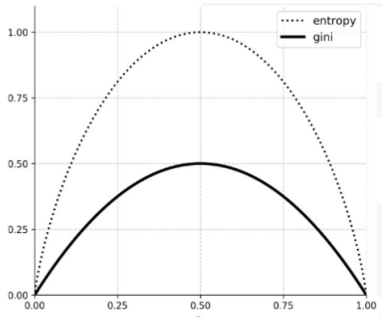
$$gain(X,fly)=entropy(X)-\frac{|X_{Yes}|}{|X|}entropy(X_{Yes})-\frac{|X_{No}|}{|X|}entropy(X_{No}) \tag{10}$$

$$\approx 0.985-(\frac{3}{7}\times 0+\frac{4}{7}\times 0.811) \tag{11}$$

$$\approx 0.521 \tag{12}$$

基尼不纯度（Gini Impurity）

基尼不纯度测量随机选择的样本如果根据标签分布进行随机标记，则错误标记的频率；衡量数据集的混乱程度。



熵之半近似等于基尼系数

$$对有k个类别的样本X：gini(X)=1-\sum_{i=1}^{k}p_i^2$$

可以作为熵的替代品来选择属性（**熵之半近似等于基尼系数，与熵的精确度大概差2%，但比熵的计算更快**）

最佳属性=最小的基尼不纯度

$$gini(X_{color=brown})=1-(\frac{1}{3})^2-(\frac{2}{3})^2\approx 0.444 \tag{13}$$
$$gini(X,Color)=p_{color=brown}\times gini(X_{color=brown})+p_{color=white}\times gini(X_{color=white}) \tag{14}$$
$$=\frac{3}{7}\times 0.444+\frac{4}{7}\times 0.5 \tag{15}$$
$$\approx 0.476 \tag{16}$$
$$gini(X,Fly)\approx 0.214 \tag{17}$$

构造决策树的算法：

ID3算法

一种基于**信息增益**的**贪心算法**，每次的节点都选择目前未被用来划分的**具有最高信息增益的属性**作为划分标准（使信息熵的下降速度最大），直到生成的决策树能完美分类训练样例。

使用信息增益来判断特征重要性程度，信息增益越大，重要性程度越大，但是其在计算类别数较多的特征的信息增益时结果往往不准确。

CART（分类与回归树 Classification And Regression Tree）

CART算法使用基尼指数衡量特征的重要性程度，基尼指数越小，重要程度越高，集可用来解决分类也可解决回归问题，CART算法效率比较高的另外一个原因是它构建的树都是二叉树，简化了树结构。

回归子regressand，回归元regressor

- CART算法是二分类常用的方法，由CART算法生成的决策树是二叉树，而 ID3 以及 C4.5 算法生成的决策树是多叉树，从运行效率角度考虑，二叉树模型会比多叉树运算效率高。
- CART算法通过基尼(Gini)指数来选择最优特征。

scikit-learn决策树算法类库内部实现是使用了调优过的CART树算法，既可以做分类，又可以做回归。有树的生成和剪枝两部分，对于树的生成采用的标准主要是：基尼系数（分类），方差（回归）；对于树的剪枝采用的标准主要是是：MCCP算法（最小代价复杂性修剪法）。

CART算法

https://blog.csdn.net/xiaqunfeng123/article/details/34820095

\TODO

正则化与修剪（Regularization and Pruning）

决策树容易过拟合,一般来需要剪枝或正则化，缩小树结构规模，缓解过拟合。正则化方法可以帮助决策树在拟合训练数据的同时，保持对未知数据的泛化能力。

以下是一些常见的决策树正则化方法（用超参数控制）：

- max_depth：**限制决策树的最大深度**：通过设定决策树的最大深度，可以限制每个叶子节点到根节点的最长路径长度，从而减少模型的复杂度。较浅的树结构通常更容易泛化到新的数据。
- min_samples_split：**一个内部节点必须包含的样本数**，这个值用来限制树的生长。如果某个节点的样本数少于这个值，则不会继续拆分该节点。为避免树过于复杂，较大的值会导致树的结构更简单，减少过拟合的风险。
- min_samples_leaf：**一个叶子节点必须包含的样本数。**如果拆分后某个叶子节点的样本数少于这个值，则该拆分不会被接受。确保决策树的叶子节点不会过于细化，从而有助于减少过拟合。
- max_leaf_nodes：**决策树的最大叶子节点数。**如果达到这个限制，树将停止生长。可以直接控制决策树的复杂度。当叶子节点数达到指定值时，算法将停止分裂节点，即使有些节点还可以继续分裂。
- max_features：**在寻找最佳划分时考虑的特征数量（可以是百分比）。**通过限制每次分裂时考虑的特征数量，可以减少模型对特定特征的依赖，提高泛化能力。

集成学习和随机森林（Ensemble Learning & Random Forests）

集成学习（Ensemble Learning）

集成（Ensemble）：一组预测器（分类器）

通过把多个基本分类器或回归器的预测结果进行合并，以得到更准确、更稳定的预测结果。集成学习可以通过多种方式进行合并，例如投票、加权平均等。投票方法将基本分类器的预测结果进行统计，选择得票最多的类别作为最终的预测结果；加权平均方法则通过给不同基本分类器分配权重，将它们的预测结果进行加权平均得到最终结果。

什么是弱学习器和强学习器？弱学习器和强学习器的主要区别体现在其性能和泛化能力上。弱学习器是指泛化性能略优于随机猜测的学习器，例如在二分类问题上精度略高于50%的分类器。其性能相对较低，效果可能低于随机猜测。相对而言，强学习器则具有很高的性能，能够在解决特定问题上表现出色。它们通常能够在训练数据和新的未见数据上都表现出色，具有较低的训练误差和较高的泛化能力。集成学习通过组合多个弱学习器，构建出一个强学习器，以提高整体性能。

(Hard) Voting Classifier

用多种（多个）机器学习方法得到的结果进行投票，**少数服从多数**。比如KNN预测结果为A，逻辑回归预测结果为A，但是SVM预测结果为B，结果进行投票是A。

Soft Voting Classifier

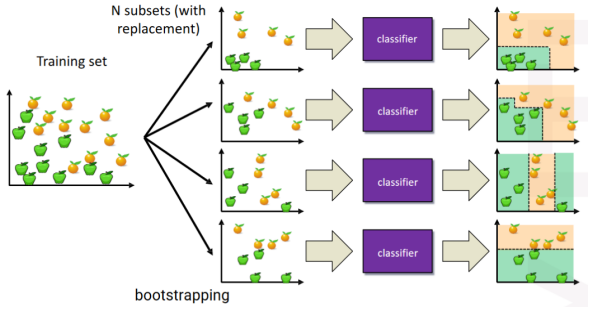
一种方法一票，少数服从多数的方法有时候是不合理的，更合理的方法应该是有权值的。如果三个分类器认为一个样本49%属于正类，两个分类器认为99%属于正类，难道应当把该样本认为是负类吗？显然不是，我们将每个分类器的概率作为权值，得到的结构更为合理。（是否还可以为不同的分类器上不同的权重）

放回取样，随机子空间（Bagging/ Pasting, Random Subspace）

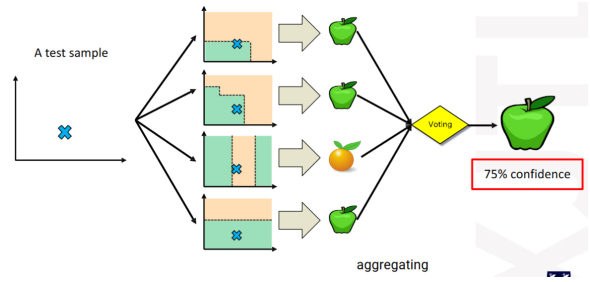
放回取样

Bagging（bootstrap aggregating**自举汇聚法**）：通过随机从训练集中有放回地抽取样本（sample with replacement），并行地训练多个独立的预测模型,并将它们的预测结果进行汇总，通过投票或平均来进行最终的决策。

训练过程：从整个训练集中每次选取一部分作为数个独立预测模型的训练集，训练完毕后放回大训练集中以供其他模型训练使用。



推理过程：样本会在数个训练完成的训练器中得到各个结果，对结果聚合，得到最终结果/可信度（confidence）



袋外评估（Out-of-Bag Evaluation）

bagging因为样本随机抽样，是基学习器没有强依赖关系，所以有降低方差的优点，除此之外，随机抽样还带来了另外的一个好处。因为bagging对训练集使用有放回的随机采样，所以会形成一个独特的袋外数据(Out Of Bag, OOB)，这部分数据是在bagging的每轮随机采样中，没有被采到的数据，可以用作测试集检测模型的泛化能力。这个袋外数据是怎么出现的呢?

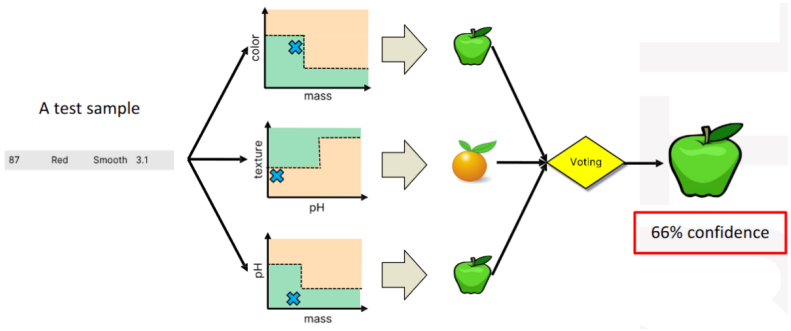
对于一个样本点，它在某一次训练集的随机采样中，每次被采集到的概率是1/m。不被采集到的概率为1−1/m。如果m次采样都没有被采集中的概率是$(1-\frac{1}{m})^m$。当$m \to \infty$时，$(1-\frac{1}{m})^m \to \frac{1}{e} \approx 0.368$。也就是说，在bagging的每轮随机采样中，训练集中大约有36.8%的数据没有被采到，这就是袋外数据。OOB的测试已被证明是无偏估计，所以在随机森林算法中不需要再进行交叉验证或者划分单独的测试集来获取测试集误差的无偏估计。

集成学习（2）bagging代表——随机森林

https://www.jianshu.com/p/9a2e5f00d613

随机子空间（Random Subspace Method, RSM）

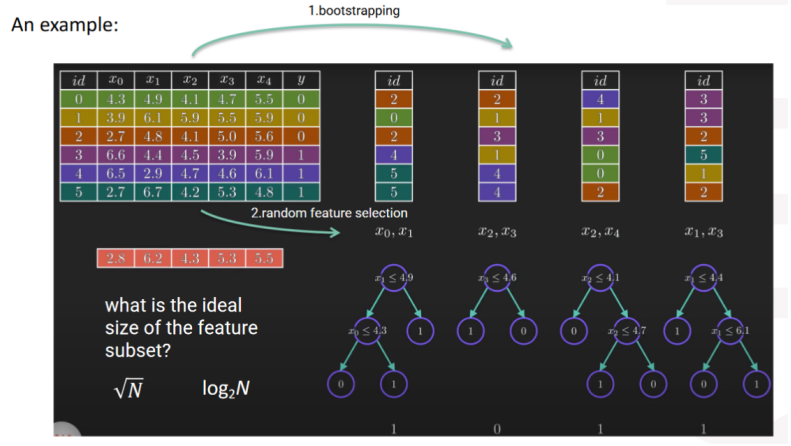
通过在训练每个分类器时使用随机的部分特征而非全部特征，来减少分类器之间的相关性。与bagging相似，bagging随机使用部分训练数据，而RSM是随机使用部分特征。特别适用于特征数量远大于训练样本数量的情况。



*同时对训练集随机采样样本和随机采样特征，被称为随机补丁方法（Random patches），可以减少决策树的方差。

随机森林（Random Forests）

随机森林结合了bagging和随机子空间的做法，不止对训练集数据进行随机采样，还对样本特征进行随机采样，这样的做法增强了基学习器之间的差异和独立性，使得集成模型的方差Variance更小，泛化能力更强，同时也增加了模型对噪声的抗性，对缺失值的敏感性降低（本身就是用缺失了特征的数据训练出来的）。

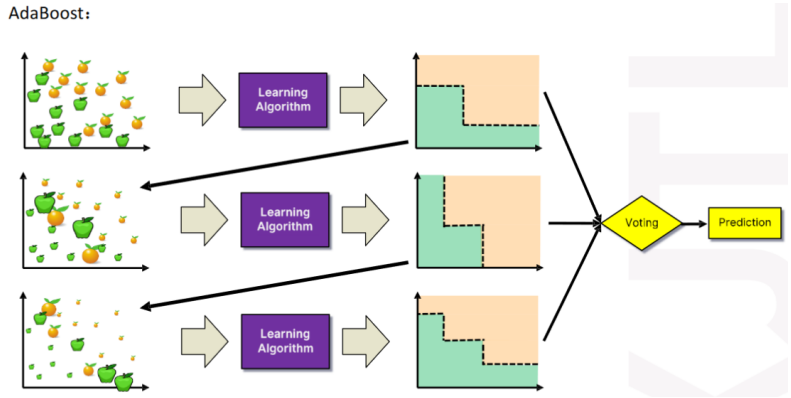


随机森林因为其天然降低方差的能力，即使单个树模型的预测对训练基的噪声非常敏感，但对于多个树模型，只要这些树并不相关，这种情况就不会出现，因此随机森林一般不会过拟合，所以随机森林中的决策树一般也**不需要剪枝**，让其自由生长即可，理论上来说决策树越多方差越小，不过树太多的话计算量会更大，所以在使用随机森林时最重要的参数就是决策树的数量，**树太少了可能欠拟合，太多了效率太低。**

提升法（Boosting）

通过组合多个弱学习器来构建强学习器的集成学习方法；各基学习器按顺序学习，迭代训练模型，同时通过增加错误分类样本的权重，使当前模型专注于先前模型的错误

AdaBoost：



梯度提升（Gradient Boosting）

Residue is learned after each iteration

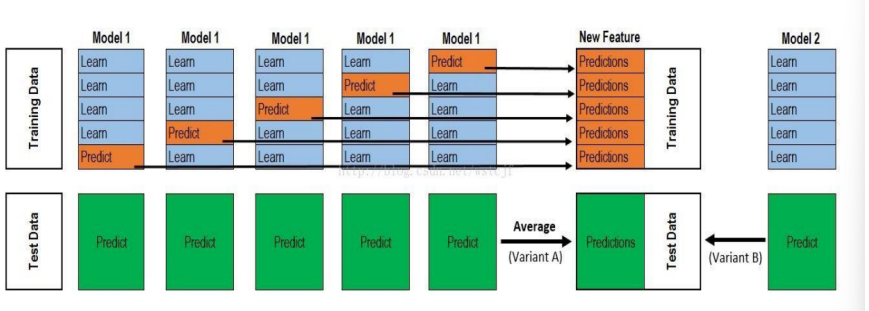
\TODO

堆叠泛化（Stacking）

通过训练一个模型来组合其他多个模型输出的方法

基本步骤如下：

- 将训练数据集划分为多个子集，通常是两个或更多个。
- 对于每个子集，使用不同的基础模型进行训练和预测，得到每个基础模型的预测结果。
- 将这些预测结果作为新的特征，组合成一个新的训练数据集。
- 使用这个新的训练数据集来训练一个元模型，例如逻辑回归、决策树等。
- 最后，使用训练好的元模型来对测试数据进行预测。



通过Stacking，不同基础模型的优势和特点可以得到充分的发挥，从而提高整体模型的性能和泛化能力。然而，Stacking也需要进行适当的调参和验证，以避免过拟合和提高模型的稳定性。

优点：

提高预测性能：通过结合多个基础模型的预测结果，Stacking可以显著提高整体模型的准确性和泛化能力。它可以充分利用不同模型的优势和特点，从而提供更准确的预测结果。

灵活性：Stacking可以集成各种类型的模型，包括线性模型、非线性模型、树模型等。这使得Stacking非常灵活，可以适应不同类型的数据和问题。

可解释性：与其他集成方法相比，Stacking在生成最终预测时使用了多个模型的预测结果。这可以提供更多的信息来解释预测结果，使得模型的解释性更强。

缺点：

计算资源和时间消耗：由于Stacking涉及到多个模型的训练和预测，因此需要更多的计算资源和时间。这可能限制了Stacking在大规模数据集和实时预测任务中的应用。

风险过拟合：如果不适当地使用Stacking，可能会导致过拟合问题。当基础模型过于复杂或训练数据集过小时，Stacking容易过度依赖训练数据，导致在测试数据上的性能下降。

超参数选择：Stacking需要选择合适的模型和超参数配置，以获得最佳的性能。这可能需要进行大量的实验和调参，增加了模型的复杂性和训练的难度。