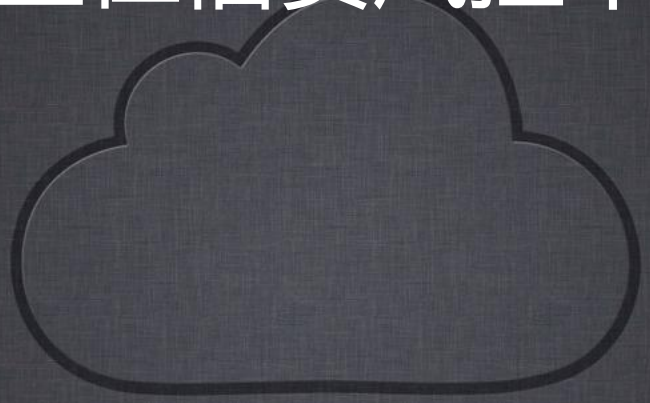
天善智能
TIANSHAN SOFT

组合模型在信贷风控中的应用



讲师：安迪生

课程简介

- 在包括违约预测在内的诸多场景中，越来越多的建模人员将模型集成的方式应用在实际工作中，并且都取得了不错的成效。本节课介绍3种基本的集成方式：bagging，boosting和stacking。

目录

元模型与集成模型

- 什么是集成模型

  “把多种单一模型组合起来共同解决一个问题”

- 集成模型的必要性

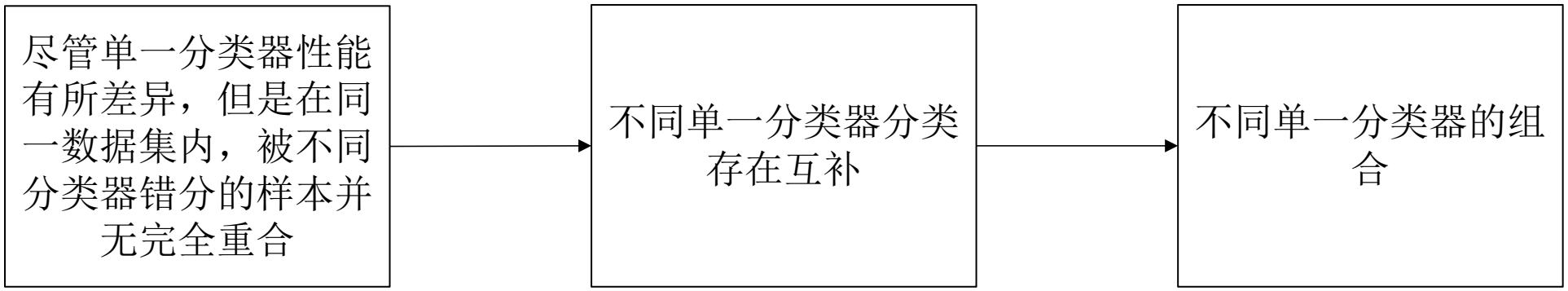
✓能够为风控模型提供更为广阔的发展空间

✓能够为风险评估的准确性、稳健性最优选择问题给出了答案

✓能够提高风控模型的效率

元模型与集成模型

- 集成模型的原理



单一分类器：基学习模型或称为元模型（base learner）

相应的算法：基学习算法（base learning algorithm）

组合方法：Boosting，Bagging&Stacking

元模型与集成模型

- 理想情况下的组合模型误差分析

考虑二分类问题$y \in \{1, -1\}$和真实函数f，假定元模型的错误率为ϵ，即对每个元模型h_i都有

$$P\big(h_i(x) \neq f(x)\big) = \epsilon$$

如果使用简单投票法，即超过半数的元模型的投票结果作为最后的结果，则有：

$$H(x) = sign\left(\sum_{i=1}^{T} h_i(x)\right)$$

在ϵ独立的前提下，有：

$$P\big(H(x) \neq f(x)\big) = \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \leq \exp\left(-\frac{1}{2} T(1-2\epsilon)^2\right)$$

元模型与集成模型

- 理想情况下的组合模型误差分析（续）

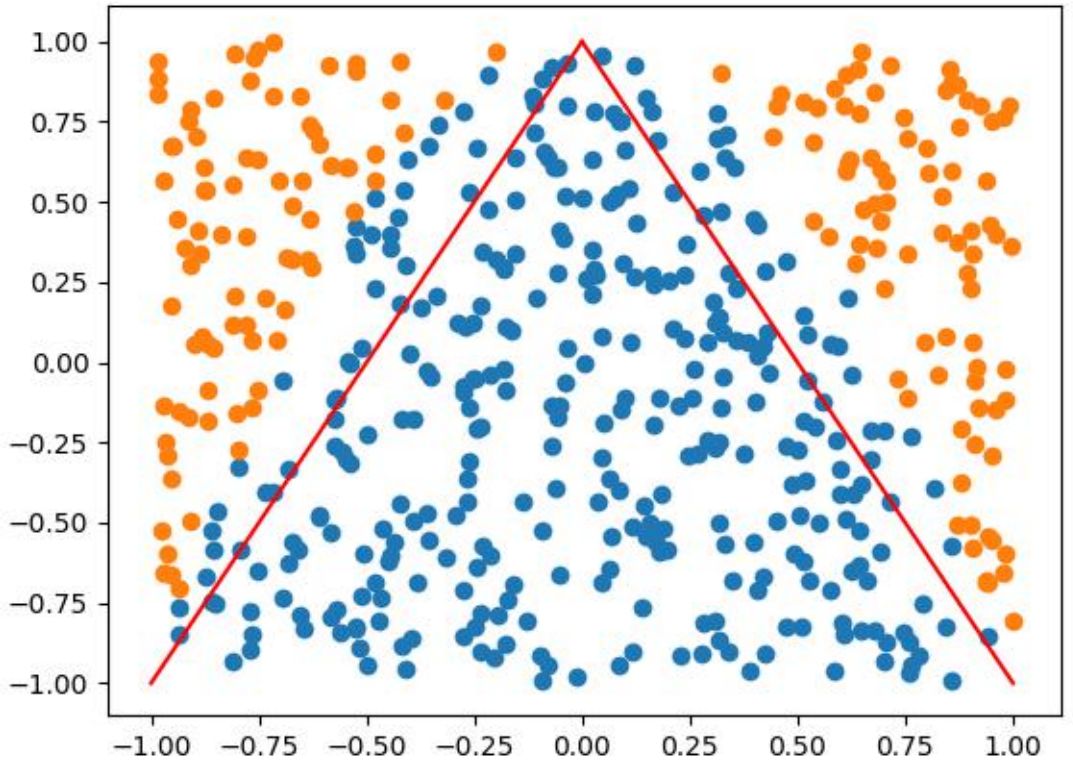
从上述的不等式可以看出：

1. 元模型的误差越小，集成模型的误差越小
2. 当元模型的误差小于0.5时，集成模型中的基分类器个数越多，则总的误差越小

然而，该理想情况一般是不成立的，原因在于最根本的假设“ϵ独立”对于同一训练集下生成的集分类器而言是无法满足的。

同时，上述结论中的要点2也非常重要。仅当元模型的误差小于0.5时（简单投票法的）集成模型才有意义。

元模型与集成模型



一个简单的例子：
红点和蓝点分属两个不同的类别，真实的判别函数是：

$$f(x_1,x_2)=\begin{cases}1, x_2<-2x_1^2+1\\ -1, else\end{cases}$$

这里我们考虑生成两个元模型，分别以两条直线表示：

$$h_1(x_1,x_2)=\begin{cases}1, x_2<-2x_1+1\\ -1, else\end{cases}$$

$$h_2(x_1,x_2)=\begin{cases}1, x_2<2x_1+1\\ -1, else\end{cases}$$

错分率均为0.25. 采取多数投票法后，错误率为0.228.

元模型与集成模型

- 集成模型中的元模型的选择

根据元模型之间的种类关系可以把集成模型划分为异态集成与同态集成两种。

异态集成

使用不同的分类、回归算法建立单一模型并进行集成

同态集成

使用同一算法(参数不同或者建立在不同的训练集上)建立单一模型并进行集成

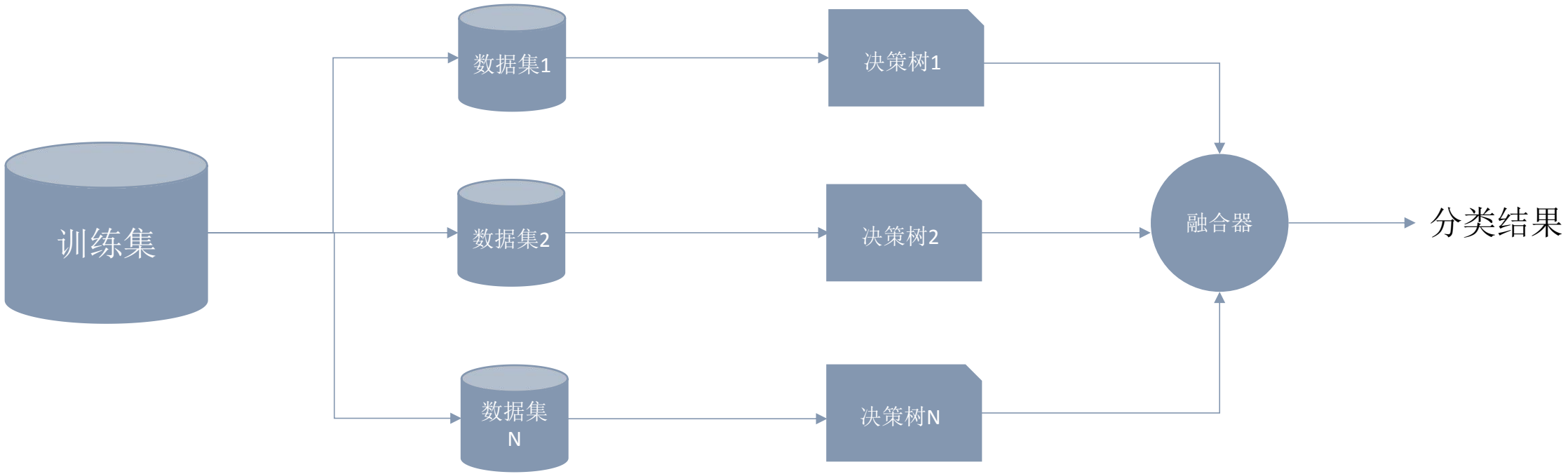
单一模型需要满足以下基本要求

- 单一模型之间的数据或者假设要求要基本相同
- 单一模型的分类错误率要低于0.5
- 单一模型之间要保证相互独立
- 单一模型的复杂度也要适度
- 单一模型的数量并非越多越好

多模型的bagging

- 集成方式一：Bagging

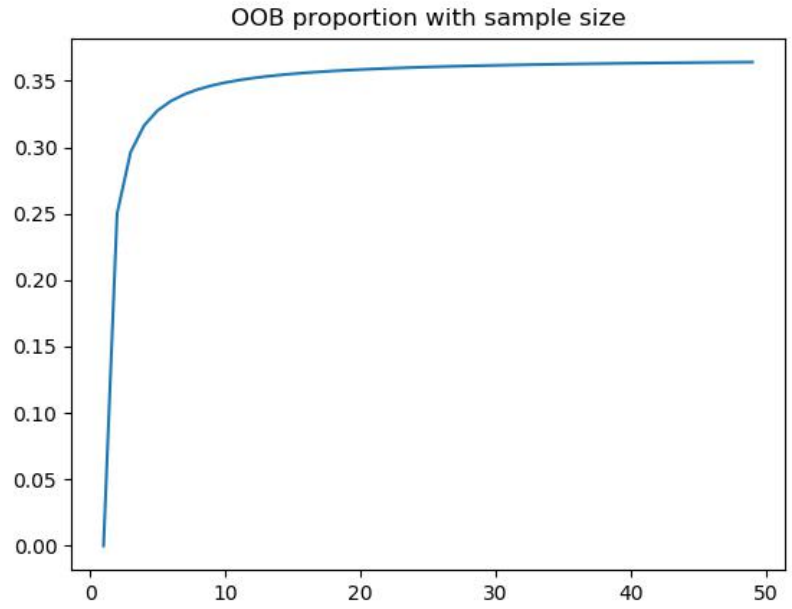
Bagging的代表是随机森林模型。这种集成方式的步骤是：



多模型的bagging

- Bagging中的数据集的构造

在Bagging集成中，需要从原训练集中有放回地抽取数据形成新的训练集，在此基础上构造元模型。假设原训练集有M个样本，则每次需要从中放回地抽取M次。由于每次抽样是随机地，因此每条样本每次被抽取到的概率为$\frac{1}{\mathrm{M}}$，每次没有被抽取到的概率为$1-\frac{1}{\mathrm{M}}$，M次都没有被抽取到的概率为$(1-\frac{1}{\mathrm{M}})^{M}$.由于$\lim_{\mathrm{M}->\infty}(1-\frac{1}{\mathrm{M}})^{M}=\frac{1}{e}\approx$36.79%, 因此当训练集较大时，约有37%的样本没有被抽取到。



没有被抽取到的数据组成的子集称为“袋外数据”（Out of Bag, OOB）

多模型的bagging

- 案例

我们使用三种模型来作为Bagging的元模型：逻辑回归模型、XGBoost模型和人工神经网络模型。对于每个模型，我们都从原始训练集中有放回地抽取样本形成同样大小的集合作为元模型的训练集。得到的每一个模型的结果是概率，求平均值后作为Bagging集成的输出。这是典型的异态集成。

在我们的案例中，元模型的AUC以及集成模型的AUC见下表。数据预处理和特征衍生参考评分卡模型的处理方法。

	LR	XGBoost	ANN	Bagging
AUC	0.762	0.795	0.879	0.893

多模型的boosting

- 集成方式二：Boosting

Boosting是另一种常见的集成方式，其基本思想是，根据当前得到的模型的错误率（或者其他与损失相关的量，例如损失函数）对样本进行调整，再构建下一个模型，最终将所有模型的结果进行加权。由此可见，与Bagging极为不同的是，Boosting中训练元分类器是串行生成的，即训练处出第i个模型后，才能训练第i+1个模型。之前介绍的GBDT和XGBoost模型就是其中一类。除此之外，还有另一个更具代表性的boosting模型：AdaBoost。它具有“可加”的结构:

$$F_M(x;P) = \sum_{i=1}^{n} \beta_i h_i(x;\alpha_i)$$

其中$h_i(x;\alpha_i)$是若干个性能逐渐提升的元模型，α_i是第i个元模型的参数，β_i是第i个元模型的权重，P是所有参数的集合。

多模型的Boosting

- AdaBoost模型（以分类场景为例）的基本思路

如何学习弱分类器

训练第i+1个元模型$h_{i+1}(x;\alpha_{i+1})$时，第i次训练中的数据集中的样本的权重需要进行调整。调整方法是：上一步中被分类错误的样本的权重得到增强，而被分类正确的样本的权重得到削弱。

初始化时，一般可以将所有训练样本的权重看成等权重，即1/样本量。

如何确定每个分类器的权重

越往后生成的分类器，其性能应该越强。但是为了增强模型的泛化能力，不能丢弃低性能的分类器。分类正确率高的分类器，其权重应该高于分类正确率低的分类器。

多模型的Boosting

- AdaBoost模型的训练步骤

1. 输入：训练数据$T = \{(x_1, y_1), (x_2, y_2), \dots, \{x_N, y_N\}, y_i = \pm 1\}$, 迭代次数M
2. 对于m=1,2,..,M
   a) 使用具有权值分布D_m的训练数据集进行训练，得到弱分类器$G_m(x)$
   b) 计算$G_m(x)$在训练数据集上的分类错误率$e_m = \sum_{i=1}^{N} w_{m,i} I(G_m(x) \neq y_i)$
   c) 计算$G_m(x)$的权重$\alpha_m = \frac{1}{2}\log(\frac{1-e_m}{e_m})$
   d) 更新训练数据集的权值分布：

$$w_{m+1,i} = \frac{w_{m,i}}{z_m}\exp\left(-\alpha_m y_i G_m(x_i)\right), z_m = \sum_{i=1}^{N} w_{m,i}\exp\left(-\alpha_m y_i G_m(x_i)\right)$$

3. 得到最终分类器

$$F(x) = sign(\sum_{i=1}^{N} \alpha_m G_m(x))$$

多模型的Boosting

- 如何解释样本权重调整准则和分类器权重调整准则

考虑AdaBoost使用的损失函数-指数损失函数具有如下的形式

$$Loss = \sum_{i=1}^{N} \exp(-y_i F_m(x_i)) = \sum_{i=1}^{N} \exp(-y_i(F_{m-1}(x_i) + \alpha_m G_m(x_i))) = \sum_{i=1}^{N} \widetilde{w_{m.i}} \exp(-y_i \alpha_m G_m(x_i))$$

其中，$\widetilde{w_{m,i}} = \exp(-y_i F_{m-1}(x_i))$是与本次迭代无关的常量。注意到，$\widetilde{w_{m,i}}$就是上一次得到的模型的损失。

对于Loss，我们有

$$Loss = \sum_{i=1}^{N} \exp(-y_i F_m(x_i)) = \sum_{y_i = G_m(x_i)}^{N} \widetilde{w_{m,i}} exp(-\alpha_m) + \sum_{y_i \neq G_m(x_i)}^{N} \widetilde{w_{m,i}} exp(\alpha_m)$$

$$= \sum_{i=1}^{N} \widetilde{w_{m,i}} \{ \frac{\sum_{y_i = G_m(x_i)} \widetilde{w_{m,i}}}{\sum_{i=1}^{N} \widetilde{w_{m,i}}} exp(-\alpha_m) + \frac{\sum_{y_i \neq G_m(x_i)} \widetilde{w_{m,i}}}{\sum_{i=1}^{N} \widetilde{w_{m,i}}} exp(\alpha_m) \}$$

由于$\frac{\sum_{y_i \neq G_m(x_i)} \widetilde{w_{m,i}}}{\sum_{i=1}^{N} \widetilde{w_{m,i}}}$是带权重意义下的分类误差率$e_m$，

$$Loss = \sum_{i=1}^{N} \widetilde{w_{m,i}} \{ (1 - e_m) exp(-\alpha_m) + e_m exp(\alpha_m) \}$$

多模型的Boosting

- 如何解释样本权重调整准则和分类器权重调整准则(续)

我们要求出α_m使得Loss最小，于是有：

$$\frac{\partial Loss}{\partial \alpha_m} = \frac{\partial \sum_{i=1}^{N} \widetilde{w_{m,i}}\{(1-e_m)exp(-\alpha_m) + e_m exp(\alpha_m)\}}{\partial \alpha_m}$$

$$= \sum_{i=1}^{N} \widetilde{w_{m,i}}\{(1-e_m)\exp(-\alpha_m)(-1) + e_m \exp(\alpha_m)\} = 0$$

即$(1-e_m)\exp(-\alpha_m)(-1) + e_m \exp(\alpha_m) = 0$

整理后有：

$$\alpha_m = \frac{1}{2} log(\frac{1-e_m}{e_m})$$

多模型的Boosting

- 案例

使用AdaBoost来建立违约预测模型。我们依然使用GridSearchCV来挑选两个最重要的参数：n_estimator 和 learning_rate，并且这两个参数同时调优。

要注意的是，AdaBoost模型对样本非平衡性很敏感。因此在建模前需要对好坏样本做平衡化的处理。我们从好样本中抽取了坏样本10倍大小的子集，与坏样本组成训练集。最终模型在全部数据集上的AUC达到0.773.

多模型的Stacking

- 多模型的堆叠（stacking）

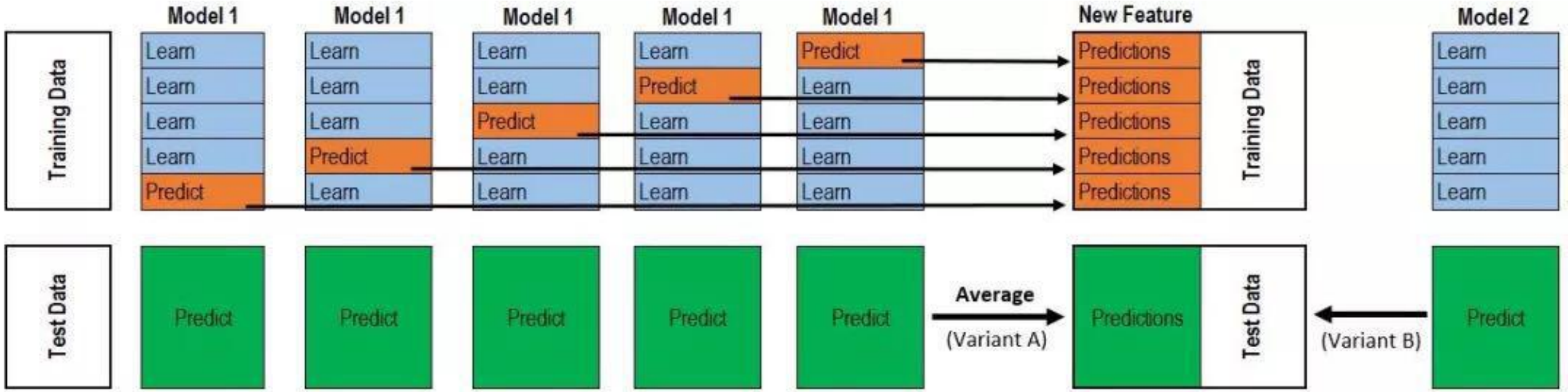
在Bagging方式的集成中，元模型的结果的集成是投票法（分类）或者平均法（回归），等价于最终的融合器是求期望。这种方式的优点是简单可靠、易解释，但是缺陷是融合器的结构过于简单，易造成信息丢失。堆叠法（stacking）改进了融合器的形式，将元模型的输入作为融合器的输出。通常元模型称为Tier1模型，融合器称为Tier2模型。当然Tier的层数可以增加，但是一般推荐2层。

除了Tier2层模型作为融合器之外，stacking的另个一改进是采用交叉验证法对训练数据集进行优化：对于训练集D，将其等分为K个子集$D_1, D_2,..., D_K$。对于Tier1中的某个模型M1，依次从$D_1, D_2,..., D_K$中挑选一个子集D_l作为测试集，剩余的子集组成交叉验证中的训练集$T_l = \mathrm{U}_{i \neq l} D_i$。M1在$T_l$上进行训练，再在$D_l$上进行预测。当$D_l$轮遍$D_1, D_2,..., D_K$后，其预测结果的并集就是数据集D的预测结果，记为$P_1$。对于模型2，3，。。。。，N，有$P_2, P_3, ..., P_N$. $P_1 \sim P_N$作为Tier2层模型的输入，进行训练。Tier2层模型的输出就是整个stacking的输出。在分类场景中，Tier2通常推荐使用逻辑会回归等。

多模型的Stacking

- 多模型的堆叠（续）

在测试时，由于Tier1的模型M_i有K次交叉验证， 因此每次交叉验证得到的模型都需要在测试集上进行预测，就得到K个结果。这K个结果需要求平均，才能得到M_i在测试集上的预测，继而带入Tier2的模型中去。



多模型的Stacking

- 案例

我们构建了Tier1模型为XGBoost+ANN、Tier2模型为LR的Stacking方式。在验证集上，XGBoost和ANN的AUC分别为表中的数据。

	XGBoost	ANN
Validation1	**0.693**	**0.649**
Validation2	**0.702**	**0.667**
Validation3	**0.726**	**0.611**
Validation4	**0.757**	**0.616**
Validation5	**0.713**	**0.633**

将验证集上的数据拼接起来形成下一层LR模型的输入，即下一层LR模型有2个输入变量。LR模型的输出的概率对应的AUC值为0.725

更多商业智能BI和大数据精品视频尽在 www.hellobi.com

BI、商业智能
数据挖掘 大数据
数据分析师
R语言 Python
机器学习
深度学习
人工智能
Hive Hadoop
Tableau
BIEE ETL
数据科学家
PowerBI

天善智能
TIANSHAN SOFT