#### 集成学习：

**1定义**：

通过构建并结合多个学习器来完成学习任务的系统。集成学习的一般结构是：先产生一组“个体学习器“，再利用某种策略将它们结合起来。



同质的算法

异质的算法

**2个体学习器之间的要求**：

1. 每个学习器具有一定的准确性；

2. 学习器之间又要有差异性

但是事实上，学习器之间的准确性和差异性是矛盾的

**3 分类**

从个体学习器的依赖关系上我们可以做如下分类：

1. 个体学习器之间存在强依赖关系，必须串行生成序列化方法，代表：Boosting

2. 个体学习器之间不存在强依赖关系，可以同时生成并行化的方法，代表：Bagging

(具体的可以参考我的博客<http://www.zhentoufei.com/detail/8/>)

**Boosting:**

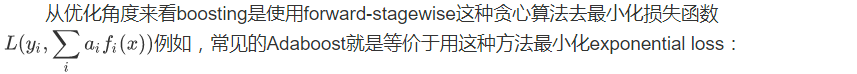
这是一种提升方法，是一种可以用来减小监督式学习中偏差的机器学习元算法。面对的问题是：一组“弱学习器”的集合能布恩那个生成一个“强学习器”？弱学习器一般是指一个分类器，他的结果只比随机分类好一点点；强学习者指分类器的记过非常接近真实值。在产生单个的基分类器的时候可以使用相同的算法，也可以使用不同的分类算法。这些算法一般是不稳定的若分类算法如BP或者决策树算法等等。最具有代表性的就是Adaboost算法了

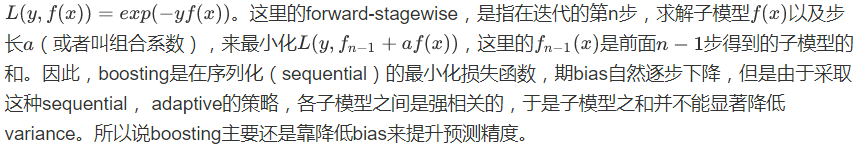
**Boosting基本思想：**

（1）先赋予每个训练样本相同的概率

（2）然后进行T次迭代，每次迭代后，对分类错误的样本加大权重(重采样)，使得在下一次的迭代中更加关注这些样本。

**Boosting降低Bias：**





关于强相关性的：极端的例子，每个模型都一样，那么肯定关于方差的改善是没有效果的。

**Boosting适用场景：**

可以用于分类和回归场景，在非平衡数据的处理上也有用武之地。在处理非平衡数据的时候只要有一个叫做BalanceCascade的算法，利用增量训练的思想，先通过下采样产生训练集，训练一个分类器，对于那些分类正确的大众样本不放回，然后对这个更小的大众样本进行下采样，训练第二个分类器，以此类推，最终组合所有的分类器结果得到最终的结果。

**Boosting特点：**

1. 降低偏差；2. 对异常值（噪声）敏感

**Bagging：**

Bagging算法（Bootstrap aggregation, 引导聚集算法），又称为装袋算法，是机器学习领域的一种团体学习算法。可以与其他分类，回归算法结合，提高准确率和稳定性的同时，通过降低结果的方差，避免过拟合的发生

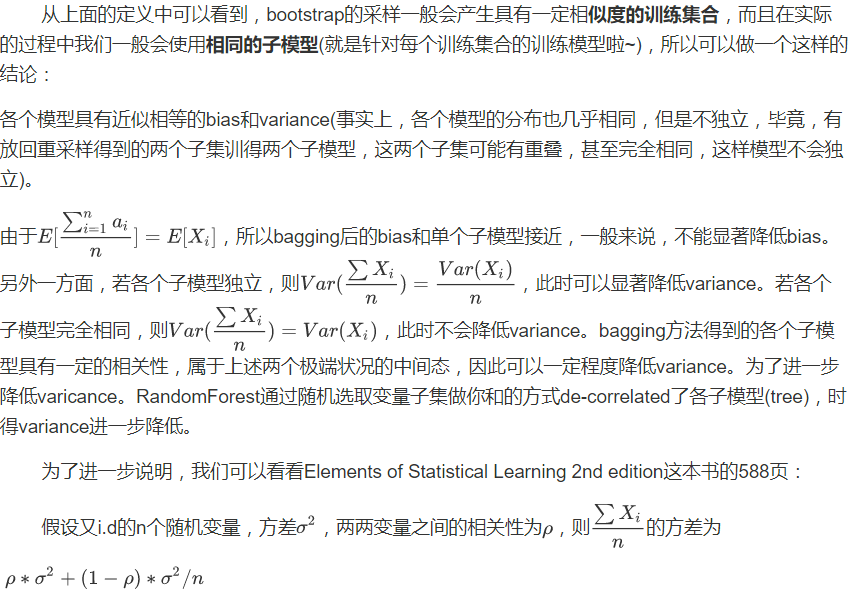
**Bagging基本思想：**

[1] 抽取：每一轮从原始样本集合中使用bootstrap的方法抽取N个训练样本（注意：存在这样的状况，有的样本被抽取到N次，个别样本一次也不会被抽取到）

[2] 训练：每次使用一个训练集合得到一个模型，N个训练集合共得到N个模型（具体的模型可以采用不同的分类或者回归方法，如决策树，感知机等）

[3] 输出：对分类问题将N个模型的输出结果采用投票的方式得到分类结果；对回归问题，计算上诉模型的均值作为最后的结果（所有模型重要性相同）

**Bagging降低方差：**





**Bagging的优势：**

1. 减少方差；2. 提高准确率；3. 抗噪声性能提升（对异常点不敏感）

**4 结合策略带来的好处**

1. 从统计方面来看，由于学习任务的假设空间往往很大，很可能有多个假设在训练集合上达到同等性能，此时如果使用单个的学习器可能因为误选而导致泛化性能不佳，结合多个学习器则会减小这一风险

2. 从计算方面来看，学习往往会陷入局部极小，有的局部极小值对泛化性能可能很糟糕，多次的晕算法之后，可以降低陷入局部最小值的风险

3. 从表示方面看，某些学习任务的假设空间有可能不在当前学习算法的假设空间中，那么，此时如果使用单个的学习器可定无效，而通过结合多个学习器，由于相应的假设空间有所扩大，有可能更好的近似



#### 模型融合

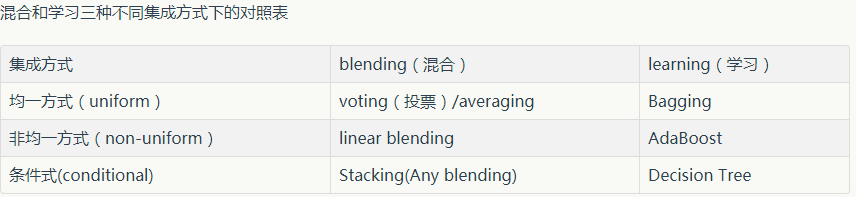
**分类：**

Uniform Blending

Linear Blending

Any Blending (Stacking)





**Uniform Blending:**

从算法的角度来看，则最常用的是采用加权型的混合推荐技术，即将来自不同推荐算法生成的候选结果及结果的分数，进一步进行组合（Ensemble）加权，生成最终的推荐排序结果。

具体来看，比较原始的加权型的方法是根据推荐效果，固定赋予各个子算法输出结果的权重，然后得到最终结果。很显然这种方法无法灵活处理不同的上下文场景，因为不同的算法的结果，可能在不同的场景下质量有高有低，固定加权系统无法各取所长。所以更好的思路是设置训练样本，然后比较用户对推荐结果的评价、与系统的预测是否相符，根据训练得到的结果生成加权的模型，动态的调整权重。

**Non-uniform Blending:**

加权混合的模型有很多，除了简单的线性模型外，常用的有回归模型（Logistic Regression）、RBM（Restricted Boltzmann Machines）、GBDT（Gradient Boosted Decision Trees），这三种混合模型在推荐算法竞赛中大放异彩，在2009年结束的Netflix百万美元推荐竞赛中，优胜队伍将充分运用和多种加权混合模型的优势，组合后的算法推荐精度非常高。获胜队的Yehuda Koren在论文The BellKor Solution to the Netflix Grand Prize中对此有非常详细的介绍。另外值得一提的是台湾大学推荐团队，他们通过混合甚至二次混合的方式（如上图），将众多单独推荐算法的结果进行最合理的加权组合，在最近几届的KDD Cup数据挖掘竞赛中所向披靡，经常取得极为优异的推荐效果。

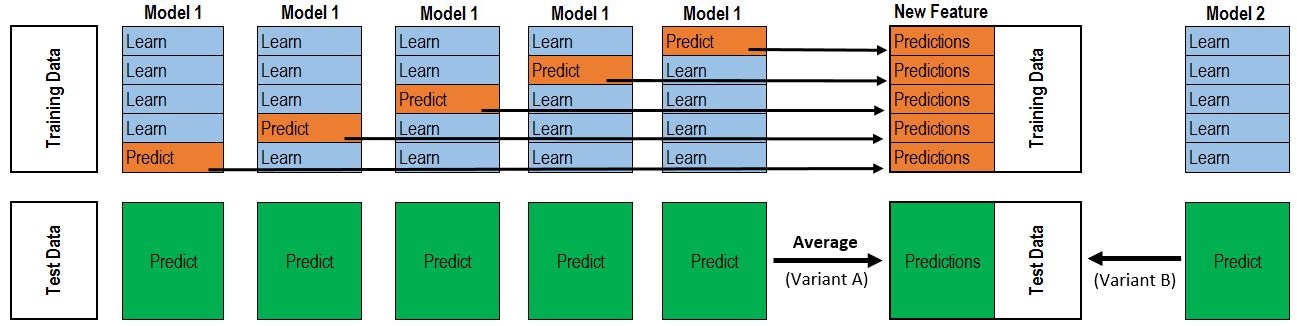
实际coding，加权可以参数可以通过线下测试集获得（需要线下与线上数据分布类似）

**Stacking：**

对于uniform/non-uniform blending的方式都比较容易理解，我们具体说说如何实现Stacking的。（感谢https://zhuanlan.zhihu.com/p/26890738）

数据介绍：train.csv有890行，也就是890人；test.csv有418行；数据的列数代表了feature（假设保留了7+1列，其中的1是预测的列）

我们先看一下stacking的流程吧



注意，这里使用了五轮的交叉训练，在五轮的训练中，我们使用的是同一个模型，这个model1都要坐满5次的训练。

Titanic的例子：

假设训练数据有890行，每一次的fold都会生成713行的小训练数据集，178行小测试数据集我们使用Model1来训练这713行的小训练数据集，然后预测178行小validation。预测的结果长度是178行的预测值

那么，这样的操作连续5次，长度位178的预测值X5 = 890个预测值，刚好和TrainData的长度吻合。这89个预测值是从Model1产生的，我们先存放起来，然后让它作为第二层训练模型的训练来源（也可以理解成新的Feature）

**Attention:**在这一步的操作得到的890X1的（890行，1列）预测值，记作P1（Predicte1）

好了，接下来我们说说TestData。在TestData中有418行（上图中的绿色的部分）。每一次的fold，713行小Train训练出来的Model1要去预测我们全部的TestData（是全部的数据！因为TestData没有做5-fold，所以每次都是全部）。此时，Model1的预测结果是418的预测值

这样的操作来5次！我们可以得到5X418的预测值矩阵。然后我们根据行来取平均值，得到1X418的平均预测值

**Attention：**这一步产生的预测值我们可以转换成418X1（418行，1列），记作p1（小写p）

到这里，Model1基本就完成了它的使命了。然后，第一层其实还是会有其他的一些模型，比如Model2，同样的走一遍，我们可以得到890X1（P2）和418X1（p2）列的预测值

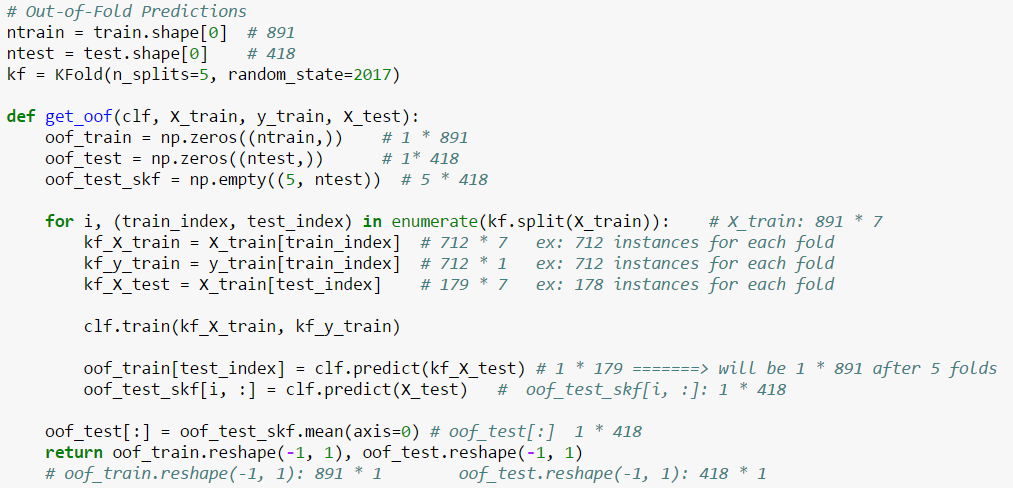
那么来自-fold的预测值矩阵890X2，(P1, P2)和来自TestData预测值矩阵418X3(p1, p2)

**来到第二层了**

来自5-fold的预测值矩阵 890 X 3 作为你的Train Data，训练第二层的模型

来自Test Data预测值矩阵 418 X 3 就是你的Test Data，用训练好的模型来预测他们吧。

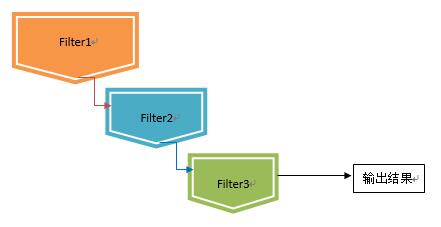
**好了，祭出大佬的PyCode**



#### 其他

瀑布融合法

瀑布型（Waterfall Model）融合方法采用了将多个模型串联的方法。每个推荐算法被视为一个过滤器，通过将不同粒度的过滤器前后衔接的方法来进行：

****

在瀑布型混合技术中，前一个推荐方法过滤的结果，将作为后一个推荐方法的候选集合输入，层层递进，候选结果在此过程中会被逐步遴选，最终得到一个量少质高的结果集合。这样设计通常用于存在大量候选集合的推荐场景上。

设计瀑布型混合系统中，通常会将运算速度快、区分度低的算法排在前列，逐步过渡为重量级的算法，让宝贵的运算资源集中在少量较高候选结果的运算上。在面对候选推荐对象（Item）数量庞大，而可曝光的推荐结果较少，要求精度较高、且运算时间有限的场景下，往往非常适用。

笔者看过天池-移动推荐大赛的最终解决方案，在样本选择时，top 5团队均选择历史用过交互的商品，而非商品全集。这也是一种Filter。

