#### 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/>)

**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

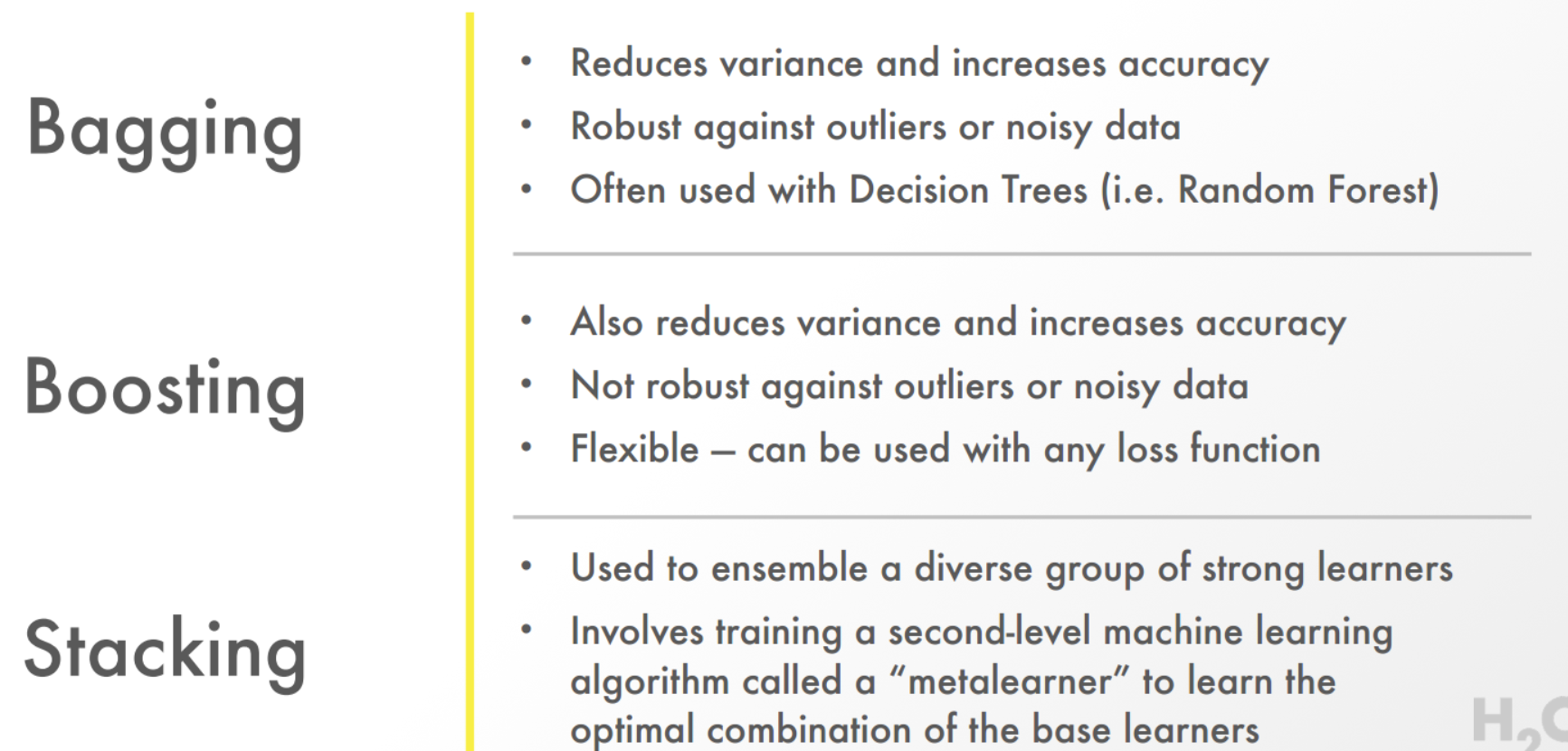
2. 从计算方面来看，学习往往会陷入局部极小，有的局部极小值对泛化性能可能很糟糕，多次的晕算法之后，可以降低陷入局部最小值的风险

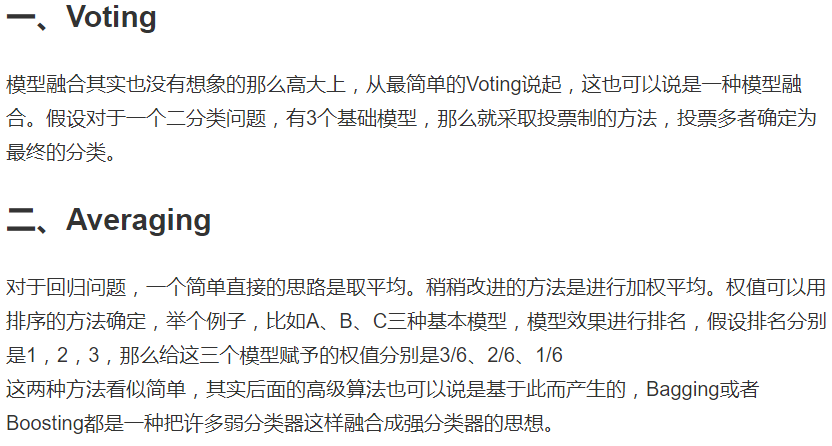
3. 从表示方面看，某些学习任务的假设空间有可能不在当前学习算法的假设空间中，那么，此时如果使用单个的学习器可定无效，而通过结合多个学习器，由于相应的假设空间有所扩大，有可能更好的近似

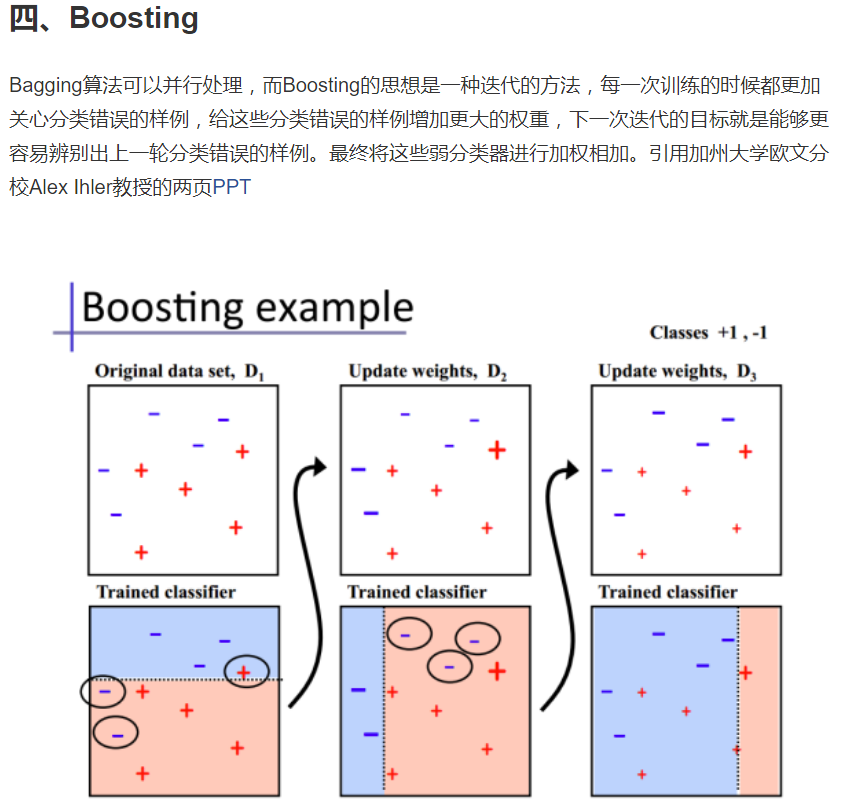
集成学习可以分为两个大类：

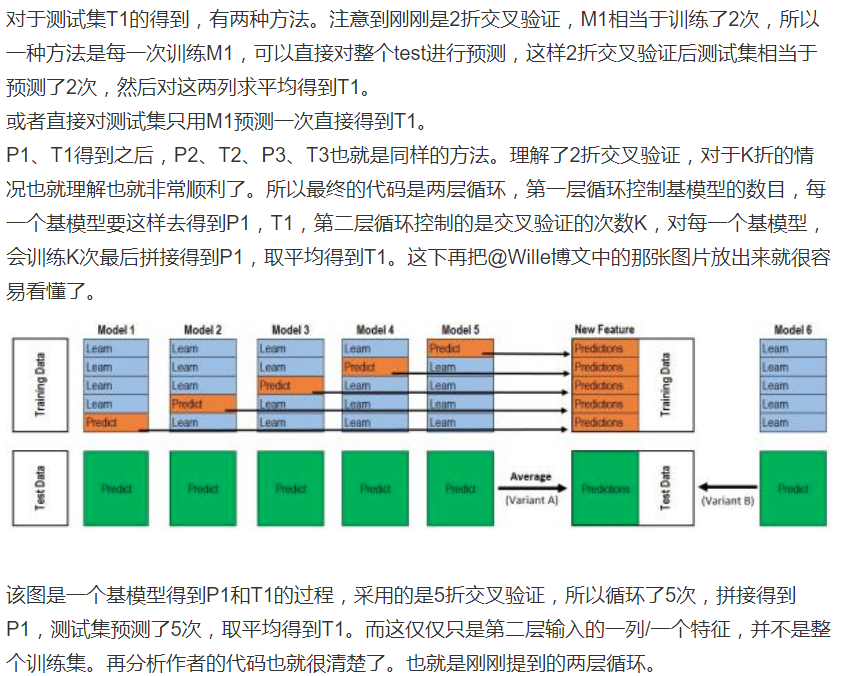
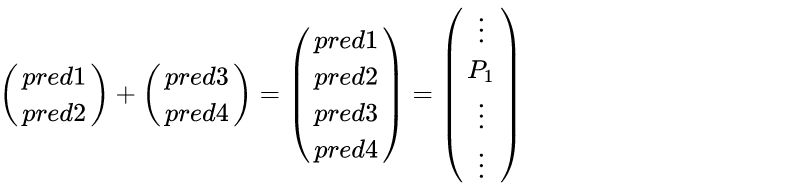
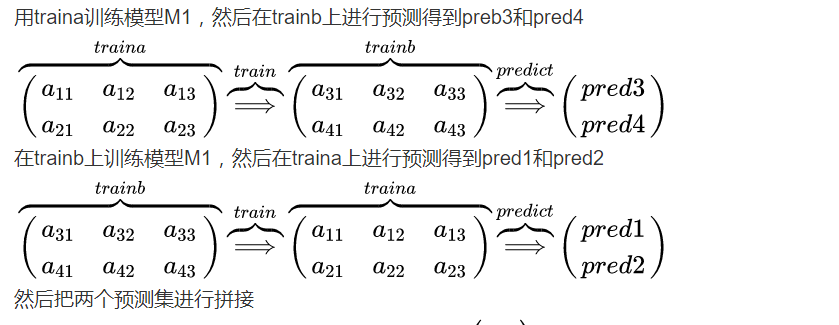
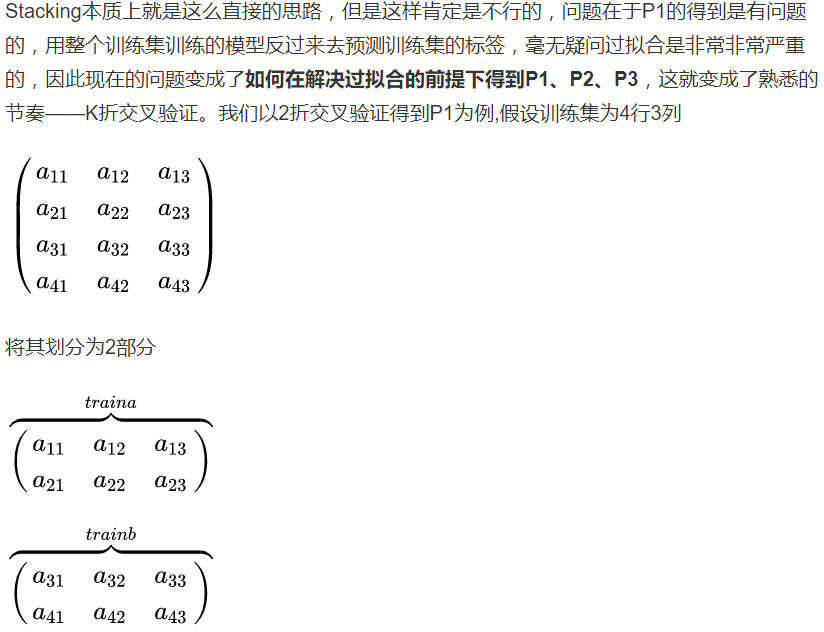
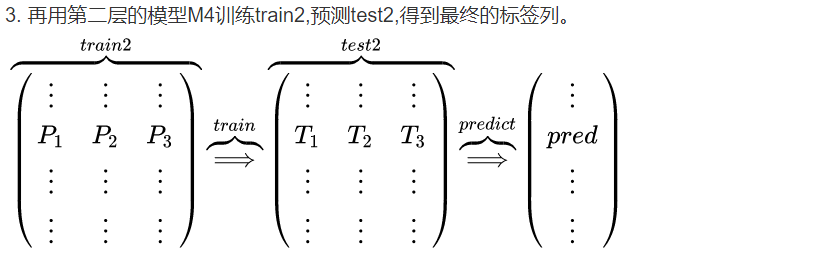
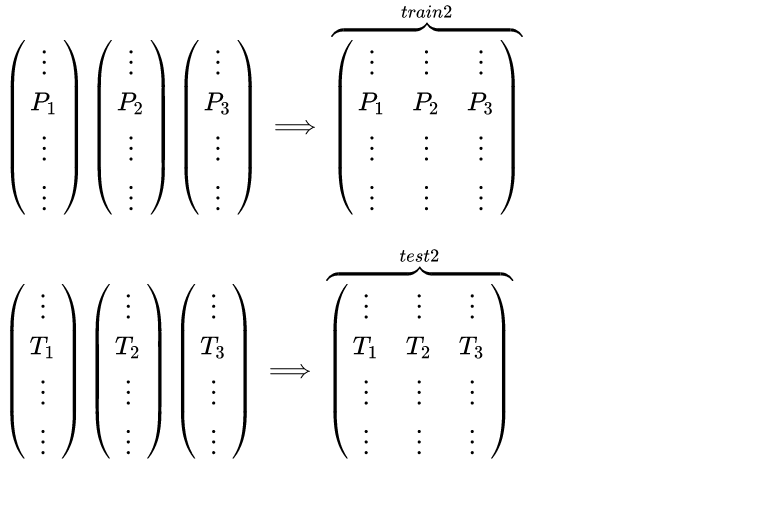
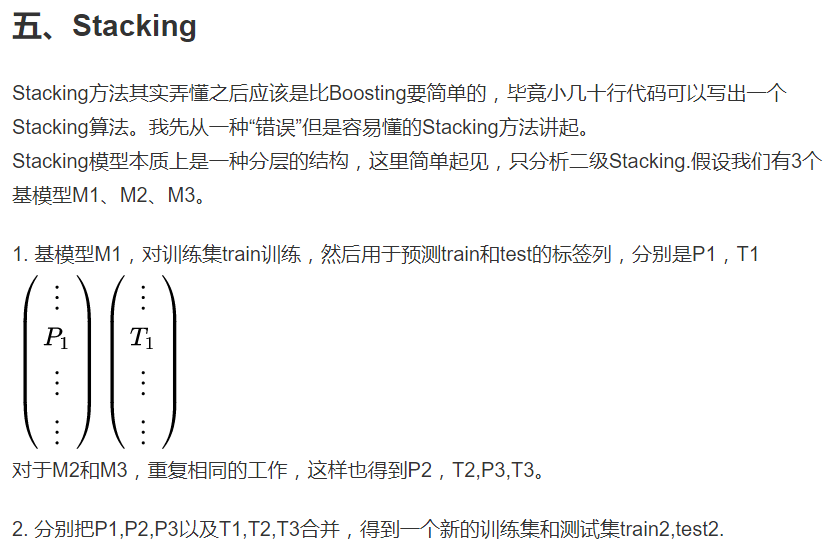
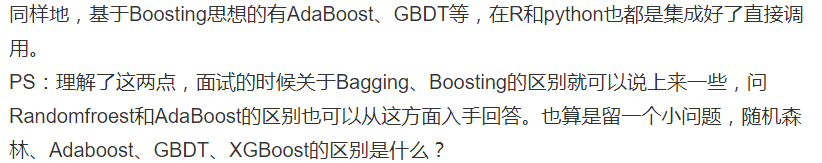
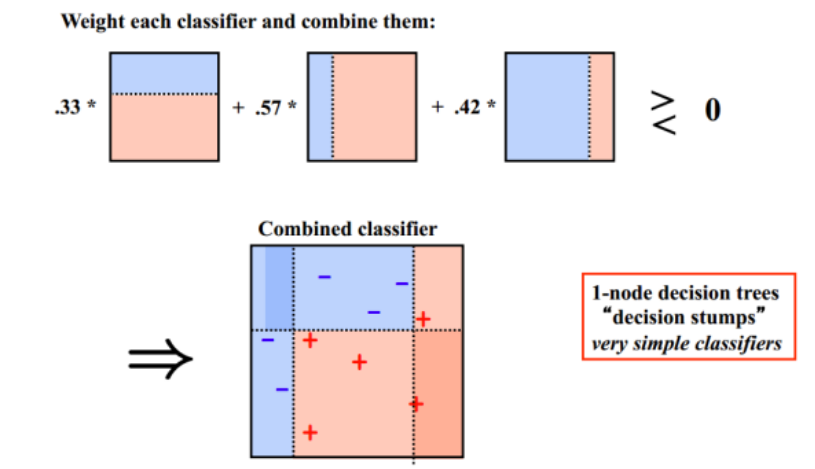
1. 模型融合

2. 机器学习元学习算法







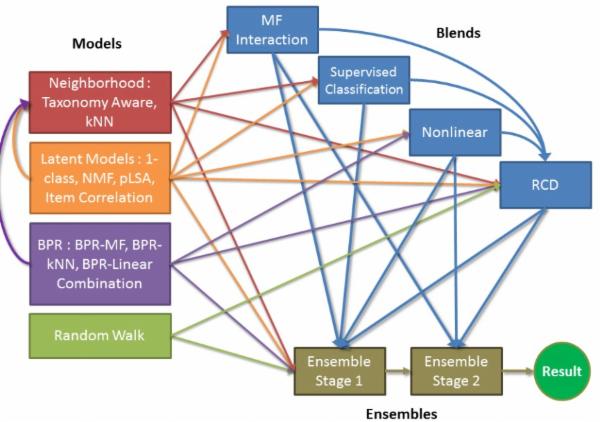


Blending：模型的组合，一般使用不同的validation set来选择不同的基础模型， 利用基础的模型进行模型的组合

1. 线性加权融合方法：

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。



**加权混合技术能明显提高推荐精度**

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。

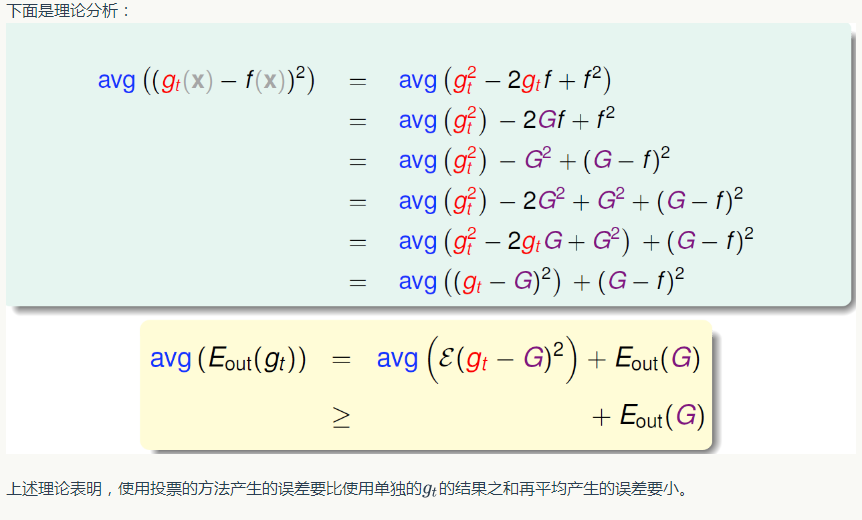
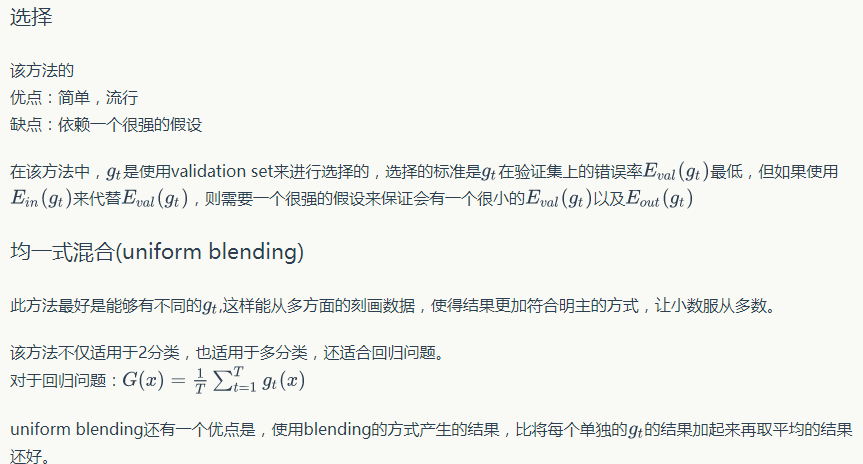
实际coding，加权可以参数可以通过线下测试集获得（需要线下与线上数据分布类似）

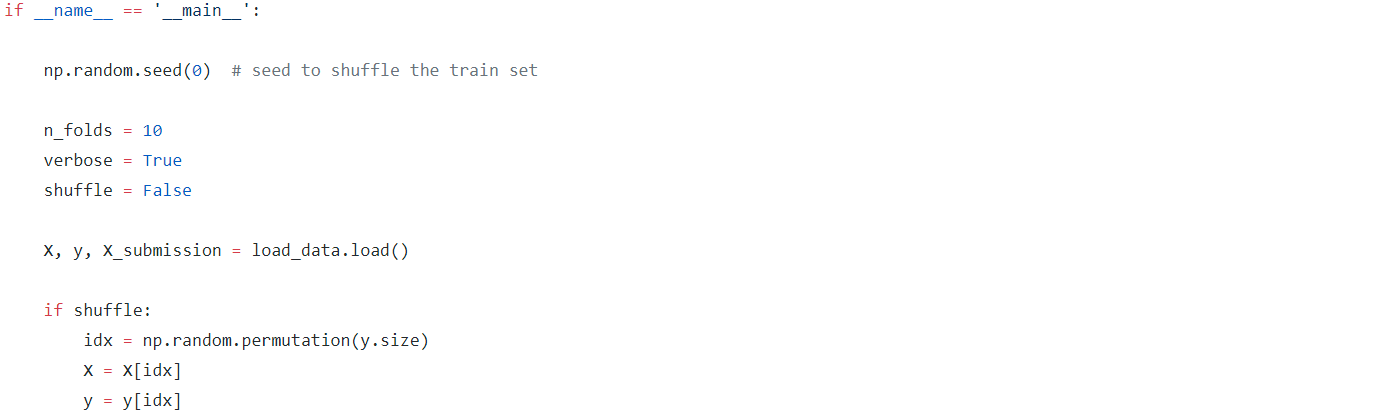
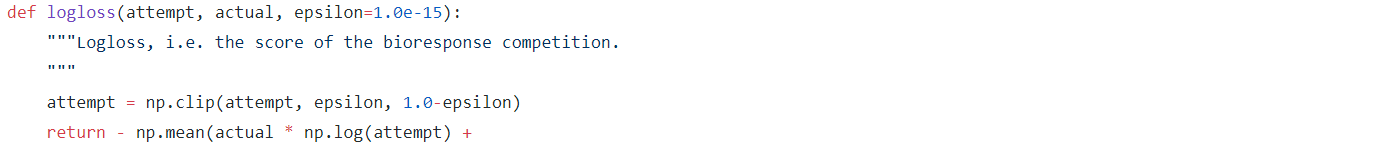
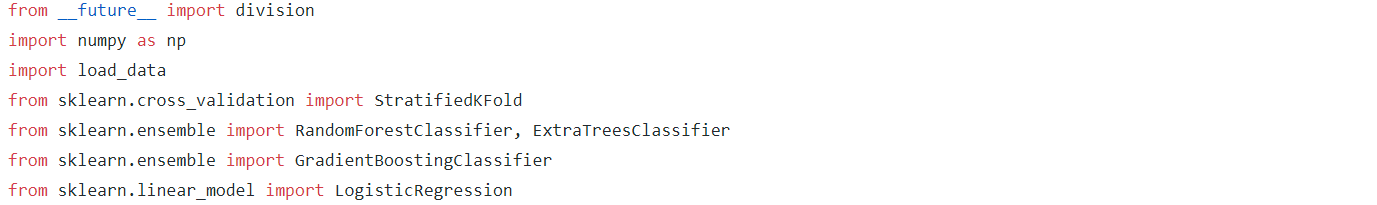
2. 交叉融合Blending

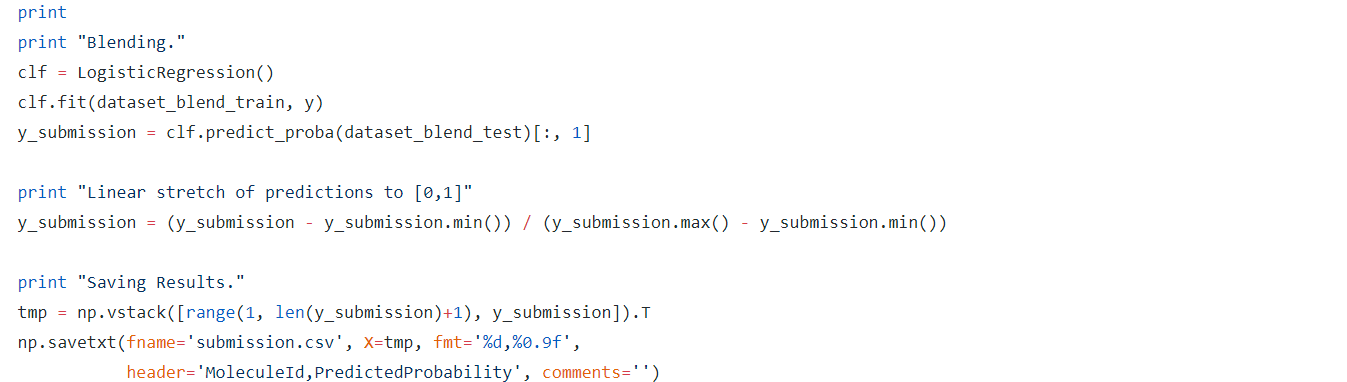
交叉融合常被称为Blending方法，其思路是在推荐结果中，穿插不同推荐模型的结果，以确保结果的多样性。

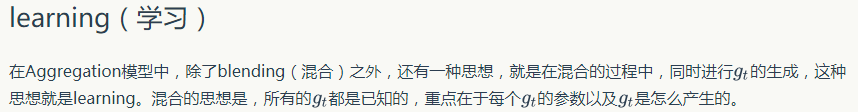
这种方式将不同算法的结果组合在一起推荐给用户

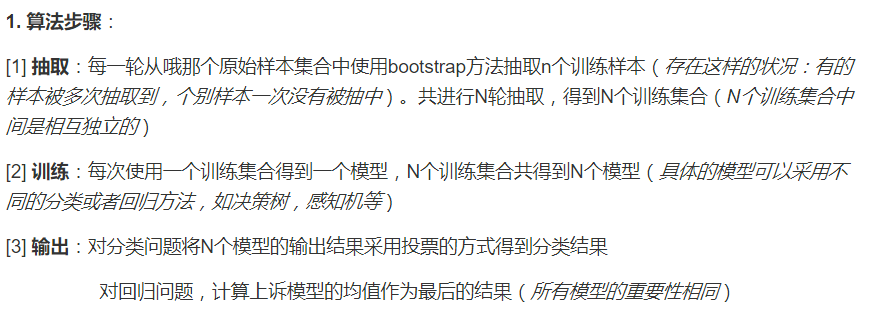
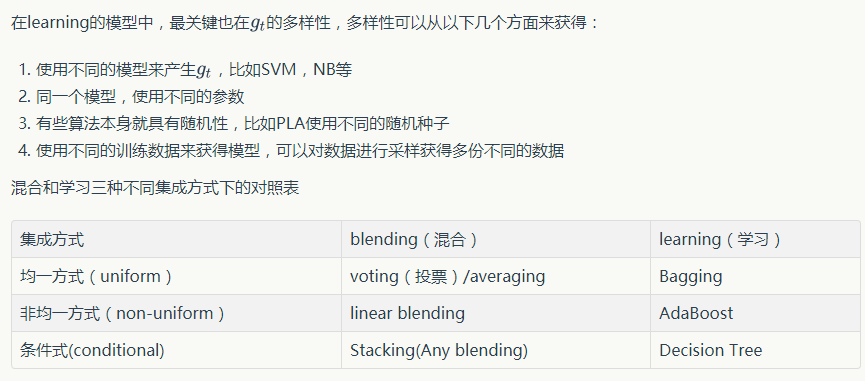
交叉融合法的思路是“各花入各眼”，不同算法的结果着眼点不同，能满足不同用户的需求，直接穿插在一起进行展示。这种融合方式适用于同时能够展示较多条结果的推荐场景，并且往往用于算法间区别较大，如分别基于用户长期兴趣和短期兴趣计算获得的结果。











例如随机森林