**集成学习**

模型融合使用的是集成学习的思想，主要分为了四种类型的集成学习，分别是bagging,boosting,stacking,blending.

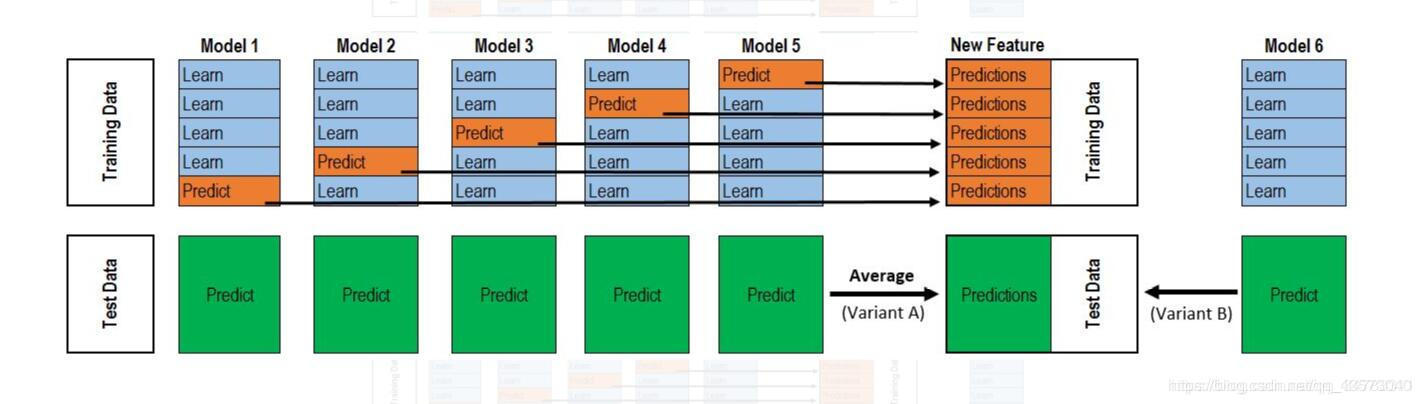
**bagging**

**有放回采样，弱学习器之间没有联系，相互独立，可以进行并行拟合**  
有放回采样:  
对数据进行随机采样(bootstrap),就是从训练集中采集固定数量的样本，但是每采集一个样本之后，都将样本放回，并且随机采样的而样本数量与训练集数量大小一致。  
袋外数据(oob)：  
由于m个样本的训练集，在每次随机采样中，被采集到的概率为1/m,不被采集到的数据概率为(1-1/m),那么经过m次采集之后，没被采集到的概率为(1-1/m)^m, 当m -> 无穷大的时候，前式是趋向于1/e, 约等于36.8%,即有36.8%的数据在m次采样后未被采集到，我们称这样的数据为袋外数据，一般用袋外数据来检测模型的泛化性能, 在sklarn库,randomclassifier()类中的oob参数就表示这个意思，默认是false,如若需要可以设置为true.  
弱学习器之间是相互独立: tbagging认为弱学习器之间的地位是相同的，是相互独立的，互相并不影响  
并行拟合:  
由于各个弱学习器之间是相互独立的，因此不每个弱学习器训练不受其他弱学习器的约束，即没有前后依赖关系，所以可以并行训练模型。  
bagging对弱学习器的选择没有限制，一般常用决策树或者神经网络  
集合策略:  
对于分类任务，一般采用的是投票选择；对于回归任务，采用加权平均方法  
bagging学习，由于每次采样不同的数据集来训练不同的弱学习器，因此泛化性能比较好，即方差较小，但是对于训练集的拟合程度会差一点，即偏差较大。  
代表性算法: 随机森林(RF)— 在bagging随机采样的基础之上，还进行了特征属性的随机采样，大大提高了模型训练的速度，以及使用CART作为弱学习器。

**boosting**

**无放回采样，弱学习器之间有联系，并不是相互独立，串行拟合无放回采样**  
Boosting思想: 采用的是加法模型，前向分步算法  
损失函数: 对于回归问题，使用平方误差损失函数，对于分类问题，使用指数函数作为损失函数  
当前强学习器是上一轮强学习器与当前弱学习器的组合，所以各个弱学习器之间并不是相互独立的，是相互影响的，所以只能串行拟合，这也是Boosting方法的缺点，在代表算法中，例如Xgboost算法就利用不同的线程实现了局部并行拟合，来提高模型训练的速度。  
Boosting 可以降低模型对训练集的拟合误差，但是训练方差较大。  
代表算法: Adaboost, GBDT, LightGBM, Xgboost,CatBoost

**stacking: 初级学习器，次级学习器**

  
(1) TrainingData进行5-fold分割，正好生成5个model，每个model预测训练数据的1/5部分，最后合起来正好是一个完整的训练集Predictions，行数与TrainingData一致。

(2) TestData数据，model1-model5每次都对TestData进行预测，形成5份完整的Predict（绿色部分），最后对这个5个Predict取平均值，得到测试集Predictions。

(3) 上面的1）与2）步骤只是用了一种算法，如果用三种算法，就是三份“训练集Predictions与测试集Predictions”，可以认为就是形成了三列新的特征，训练集Predictions与测试集Predictions各三列。

(4) 3列训练集Predictions+TrainingData的y值，就形成了新的训练样本数据；测试集Predictions的三列就是新的测试数据。

(5) 利用meta model（模型上的模型），其实就是再找一种算法对上述新数据进行建模预测，预测出来的数据就是提交的最终数据

**blending**

算法简单思想就是: 假如总的数据集为12500条，其中训练集training data的条数为10000，测试集testing data的条数为2500；然后将traing data 分为7000和3000，然后使用m个模型来训练7000条数据，然后再将m个模型在3000条数据上进行预测，将m个模型预测的结果进行拼接，则会得到3000 x m的数据，同时对testing data进行预测，将预测结果与3000 x m 合并共同作为第二层training data.最后再使用一个模型来进行预测即可。(和stacking思想有点像，但是区别还是挺大的。)  
主要优缺点如下：  
优点:  
比stacking简单(因为没有进行交叉验证来获取新的feature)  
避开了信息泄露问题  
缺点：  
使用了很少的数据  
blender可能会过拟合  
stacking使用了多次的CV会比较稳健。

*图片及部分内容为引用*