机器学习训练营第六章作业

一期 何文迁

目录

[一、 准备和分析](#_Toc617536662_WPSOffice_Level1) [3](#_Toc617536662_WPSOffice_Level1)

[二、 调试操作](#_Toc682664835_WPSOffice_Level1) [4](#_Toc682664835_WPSOffice_Level1)

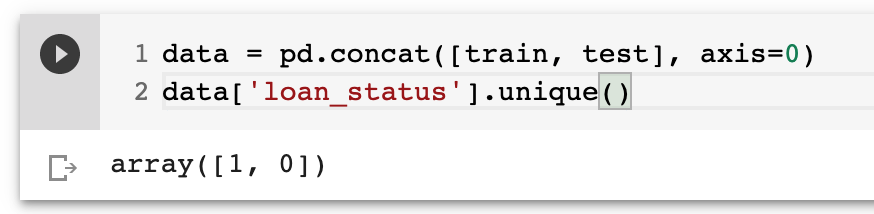
[三、 总结和结果](#_Toc1690239571_WPSOffice_Level1) [19](#_Toc1690239571_WPSOffice_Level1)

1. 准备和分析

首先分析和计划准备的调参步骤，按照课堂笔记首先计划测试的参数是三个基础参数:num\_leaves，num\_round和learning\_rate，计划选择一个num\_leaves和learning\_rate联合效果最好的组合，而num\_round则根据效率和收敛树棵数使用合适的值。然后原计划准备接着测试feature\_fraction和bagging\_fraction，但在查阅资料后认为先对树涉及的参数调整会好一些，所以先测试了参数min\_data\_in\_leaf，min\_sum\_hessian\_in\_leaf和max\_bin。接着测试feature\_fraction，bagging\_fraction及关联的bagging\_freq。最后再根据表现观察，如果还认为有提升空间则尝试正则化的参数lambda\_l1和lambda\_l2。直到发现新参数对结果影响逐次减小，认为继续测试新参数对误差优化的程度不会很高，则终止测试将当前的结果作为本次作业的输出提交。

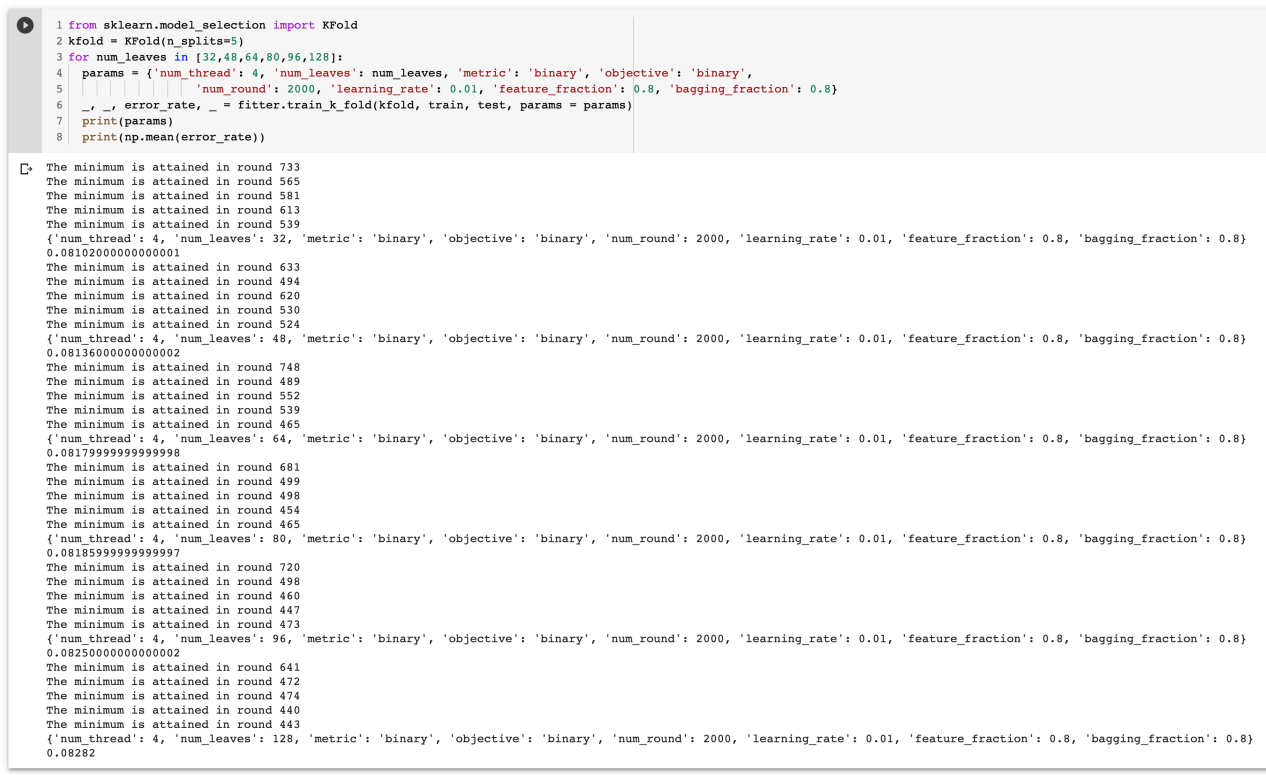
1. 调试操作
   1. binary

首先看下load\_status的样本表现，发现这里的值也是离散的1和0所以objective直接确定是binary了



* 1. num\_leaves、num\_round和 learning\_rate

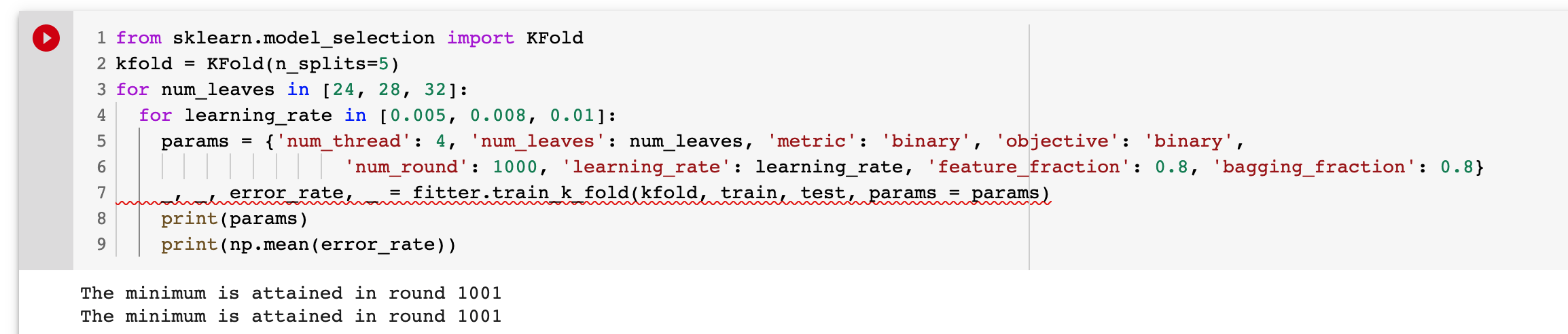
接着老师上课使用的几个参数，测试一下不同的树深表现，结果如下:

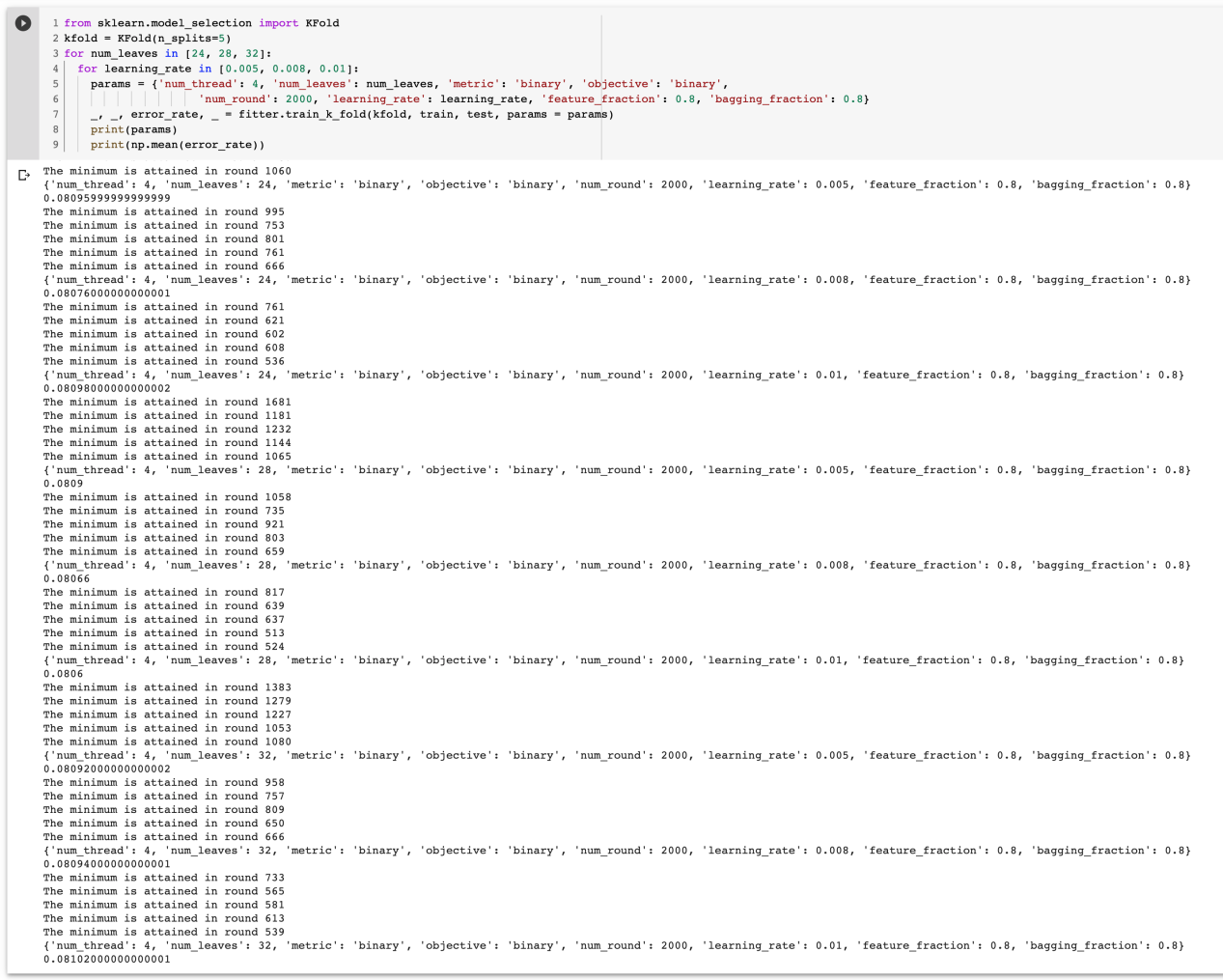


本次测试最小值为0.08102。从上可以得到几个结论:

* + 1. 整体上树越深误差越大，所以接下来测试的树深应小一点；
    2. 树深翻几倍测试后虽然误差有差别但是差的都不是很大，估计整个数据集中没有特别强的变量，
    3. 收敛的树序号都比较小，接下来准备尝试更小的学习率

调整：在此基础上将num\_round调整为1000，然后依次测试几个不同的树深和学习率观察结果。这里故意保留上次测试表现最好的num\_leaves:32和 learning\_rate:0.01目的是观察新的测试参数下的相对效果。结果发现收敛的树棵数大于1000，直接终止调整num\_round，逐次增大到 2000才获得预期的效果，过程如下:



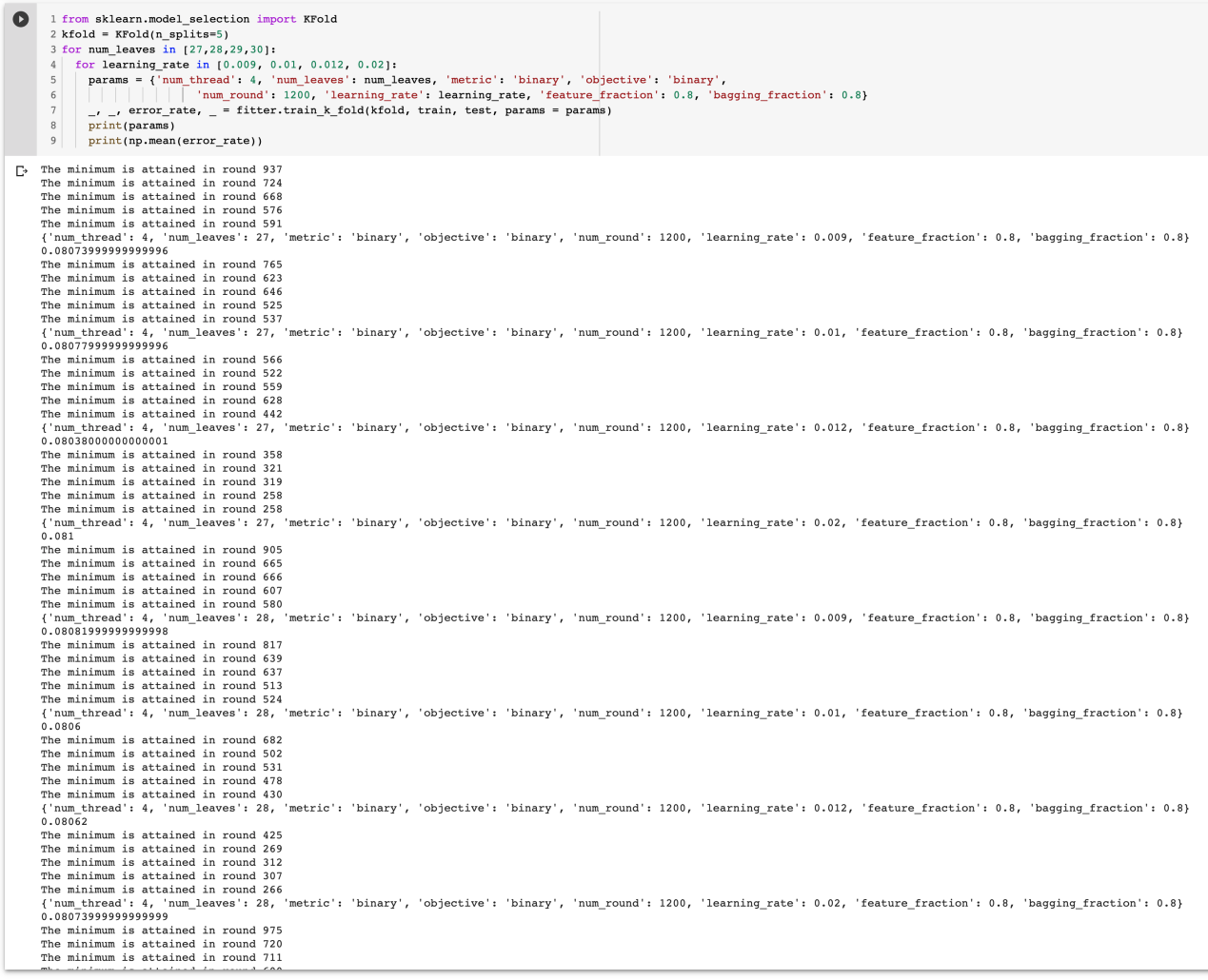


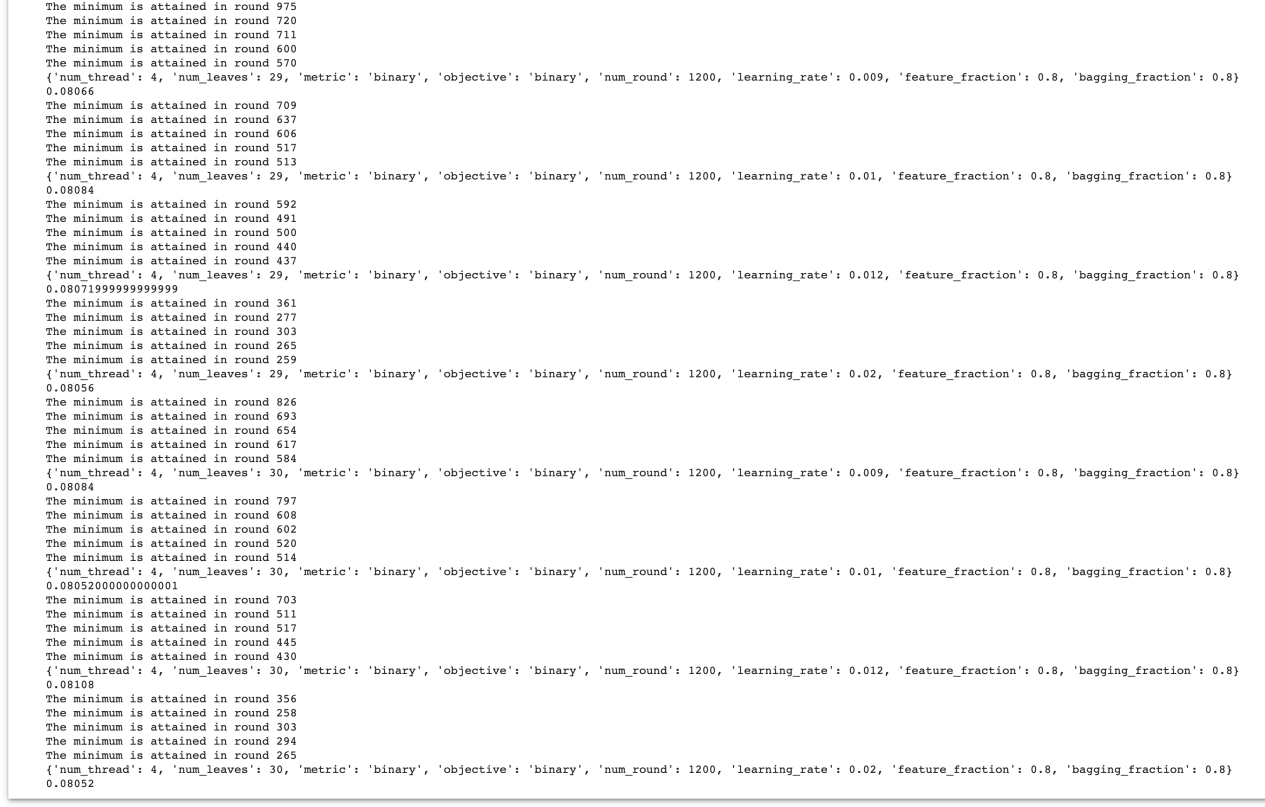
通过本轮测试发现目前最小误差来到了0.0806，由参数组合{'num\_thread': 4，'num\_leaves':28，'metric':'binary'，'objective':'binary'，'num\_round':2000，'learning\_rate':0.01，'feature\_fraction':0.8，'bagging\_fraction':0.8}给出，收敛的树棵数大约在600左右。据此分析得知：24和32的叶子节点数都没有28好，接下来应着重围绕28尝试找更好的组合

而学习率方面最优的误差由0.01得出，同时看到0.08学习率在其它参数相同情况下的误差0.08066也比较接近，那么准备尝试下0.09以及比0.01稍大的学习率。

树棵数由于较好的误差收敛都比较快，结合接下来准备的微调预估，修改为1200节省时间。

所以接下来准备测试的参数值为:learning\_rate测试0.009，0.01，0.012，0.015，0.02值，主要看看0.01附近的表现，也顺便看下差距较大的0.02会不会有意料之外的好表现(预计是比较差的)；num\_leaves测试28附近的值(主要测试稍大的num\_leaves)，取num\_leaves为27，28，29，30；num\_round基于上述两个参数的选择取1200，测试结果如下:



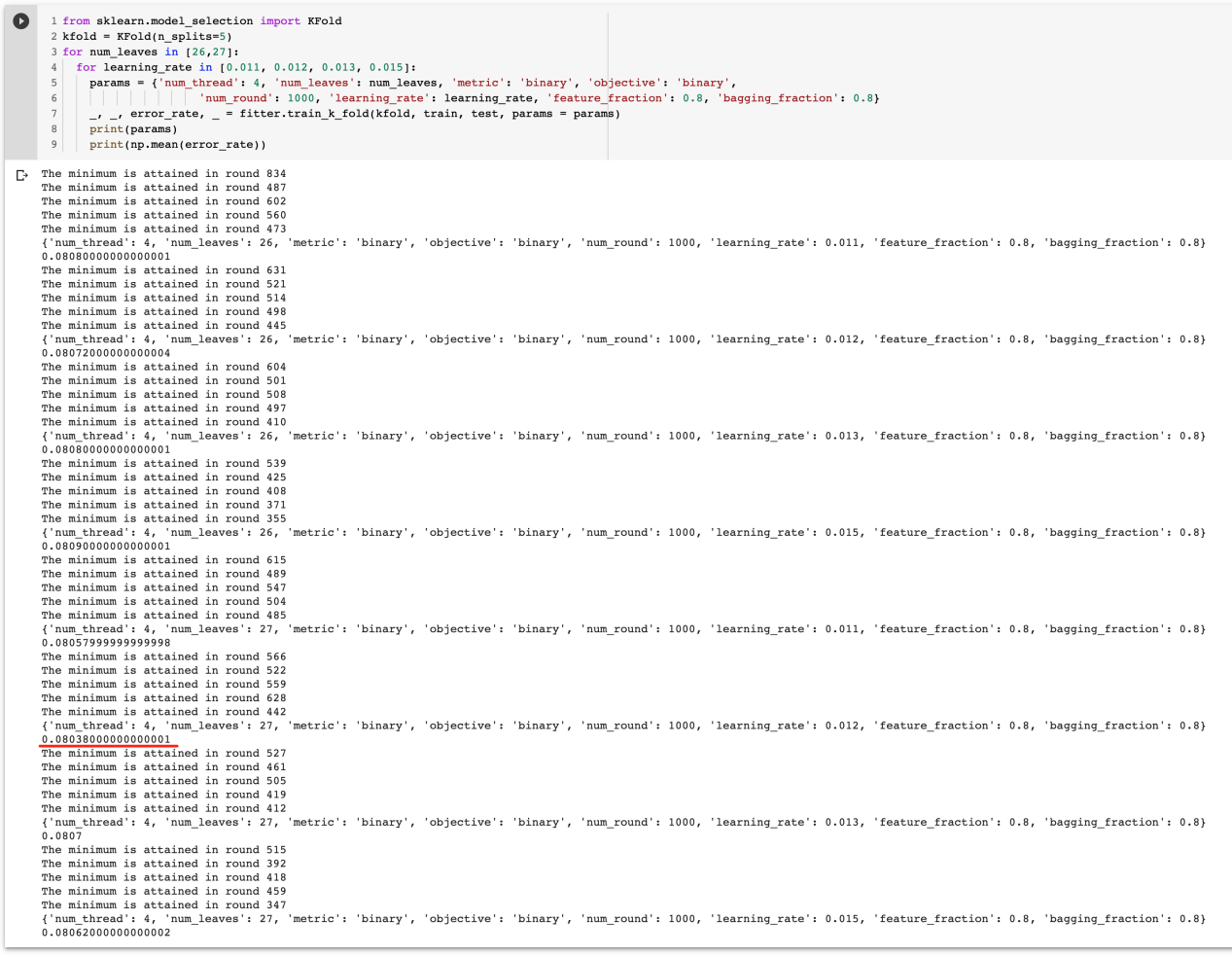


分析：本轮最好表现是由{'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1200，'learning\_rate':0.012，'feature\_fraction':0.8，'bagging\_fraction':0.8}参数给出的结果0.08038000000000001。跟预期不一样的是最优num\_leaves反而在27附近得到，而学习率也表明还有继续深化的必要性。

同时在{'num\_thread': 4，'num\_leaves': 30，'metric':'binary'，'objective':'binary'，'num\_round':1200，'learning\_rate':0.01，'feature\_fraction':0.8，'bagging\_fraction':0.8} 和 {'num\_thread': 4，'num\_leaves': 30，'metric':'binary'，'objective':'binary'，'num\_round':1200，'learning\_rate':0.02，'feature\_fraction':0.8，'bagging\_fraction':0.8}也得到0.08052000000000001和0.08052两个相对不错的结果，计划在这两个方向都测试下观察相对的效果是不是有更好的。

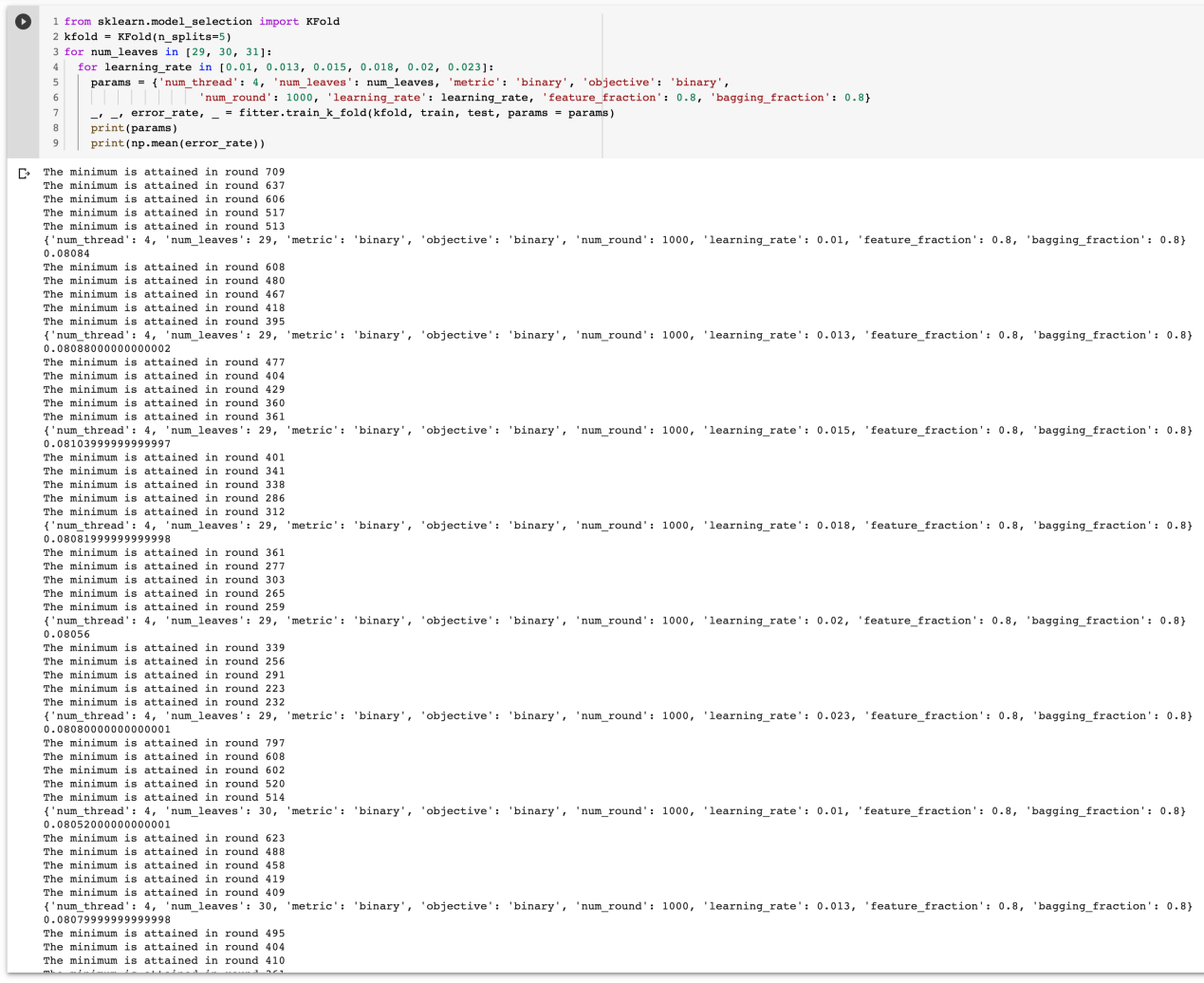
首先针对误差0.08038000000000001调参，围绕num\_leaves在26和27之间选择，同时搜索学习率在0.012附近是否有更好的值，而num\_round的话观察下来1000已经足够。

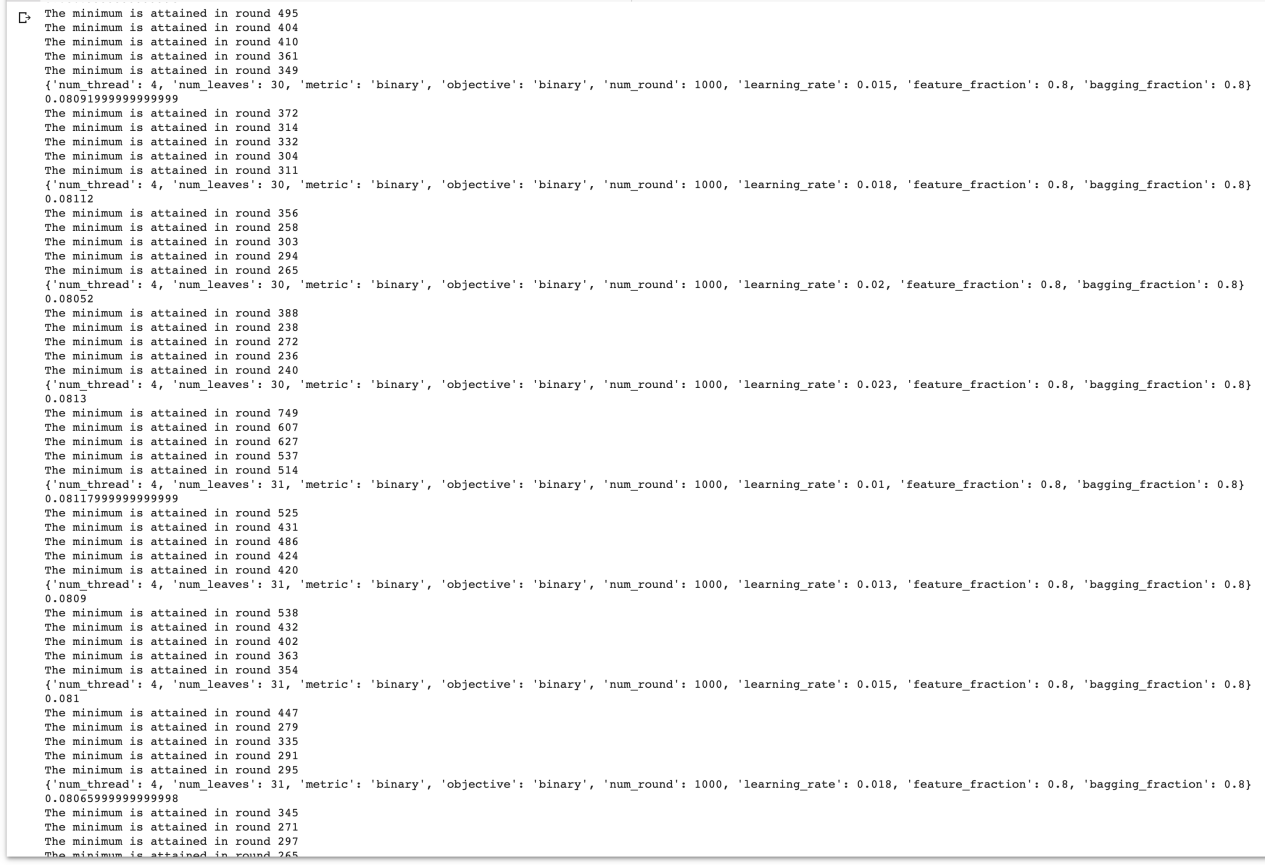
测试结果如下：

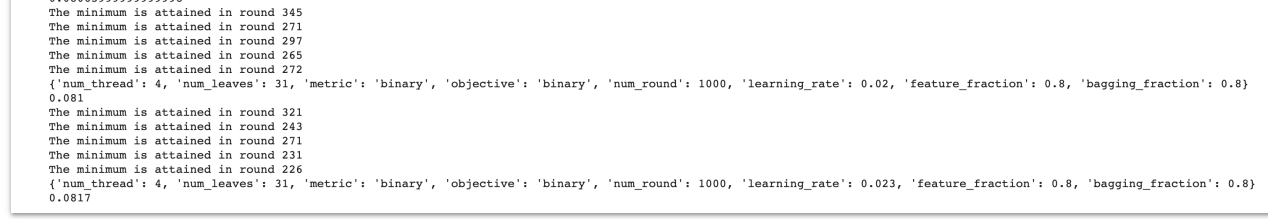


本轮测试验证了在当前条件下num\_leaves在27取得最优，而 learning\_rate则在0.012表现最好。

最后尝试性看下上面0.08052000000000001和0.08052附近有没有更好的参数组合，如果没有则认为0.08038是目前三个参数调试能找到的最好的结果。探索性测试的结果如下：





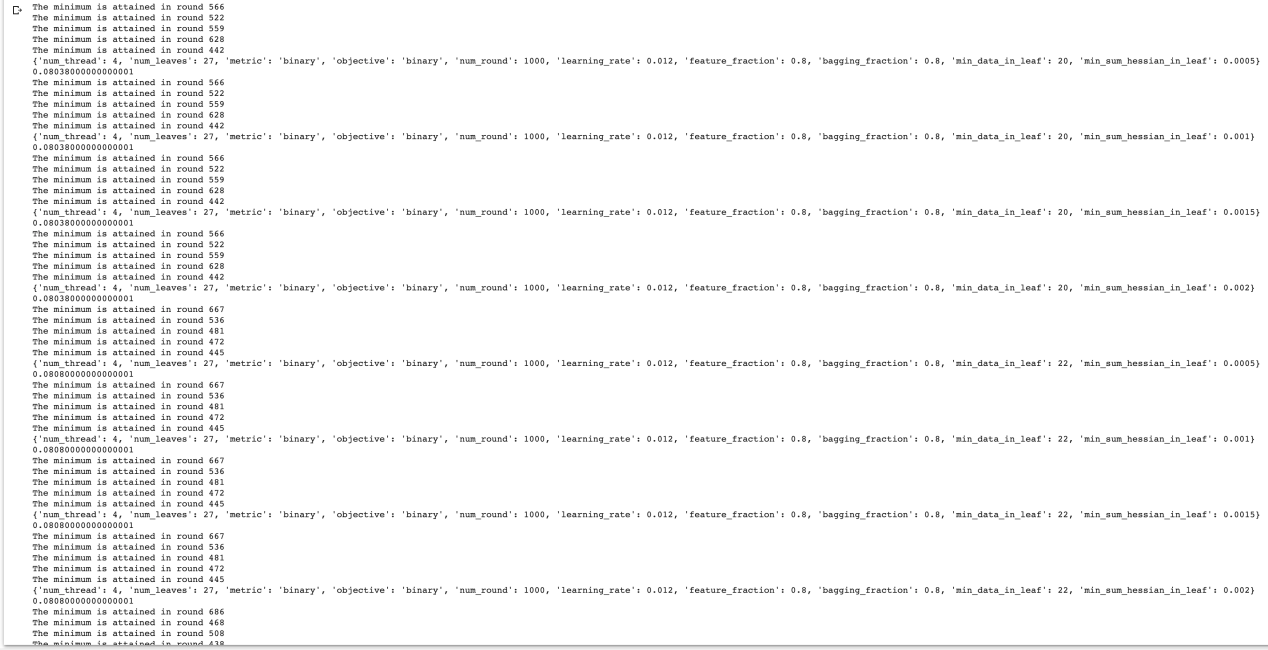


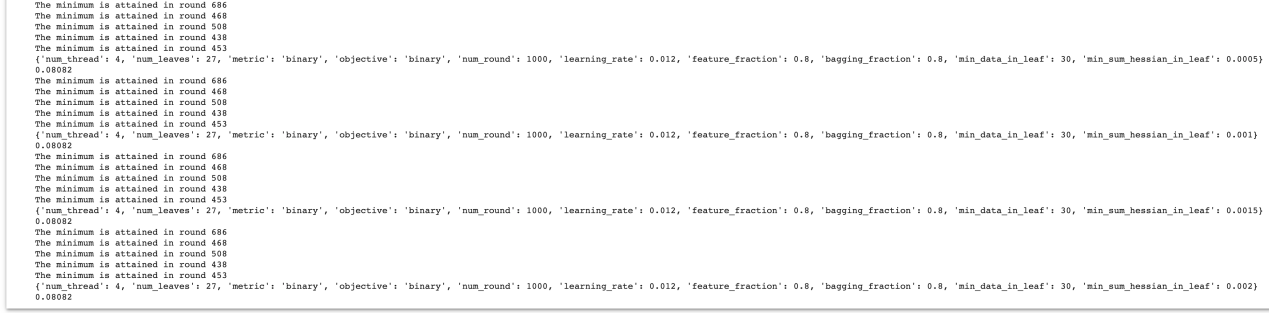
截图表明并没有找到比0.08052更好的结果，所以针对num\_round，num\_leaves，learning\_rate三个参数的调试阶段性结果是0.08038000000000001,由参数组合{'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1000，'learning\_rate':0.012，'feature\_fraction':0.8，'bagging\_fraction':0.8}给出

* 1. min\_data\_in\_leaf，min\_sum\_hessian\_in\_leaf和max\_bin

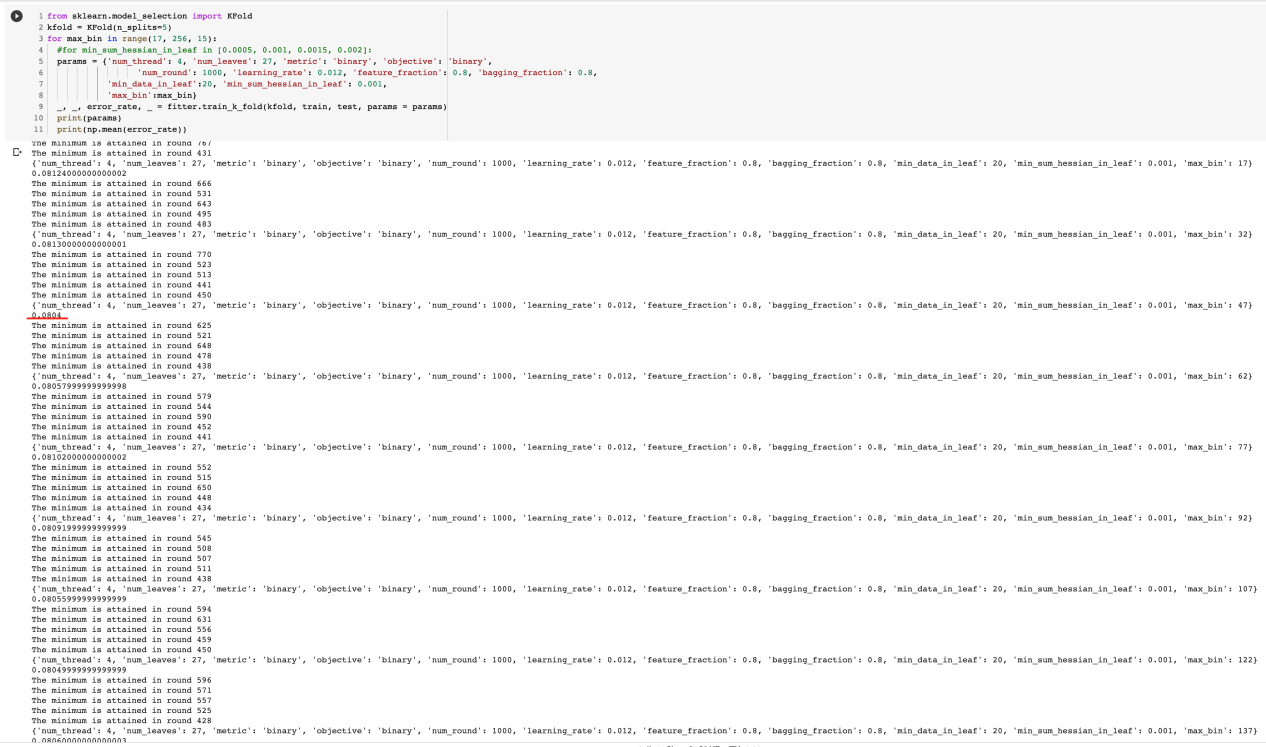
本来准备接着对 feature\_fraction和bagging\_fraction进行调参，但是查阅资料后发现通常都是先对树涉及的参数调整，所以先测试优化min\_data\_in\_leaf，min\_sum\_hessian\_in\_leaf和max\_bin。首先根据默认值周边的范围搜索一下，结果如下图:



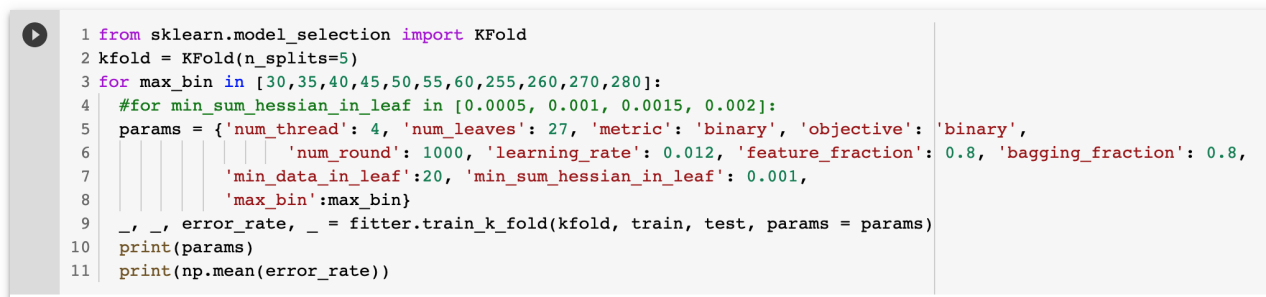




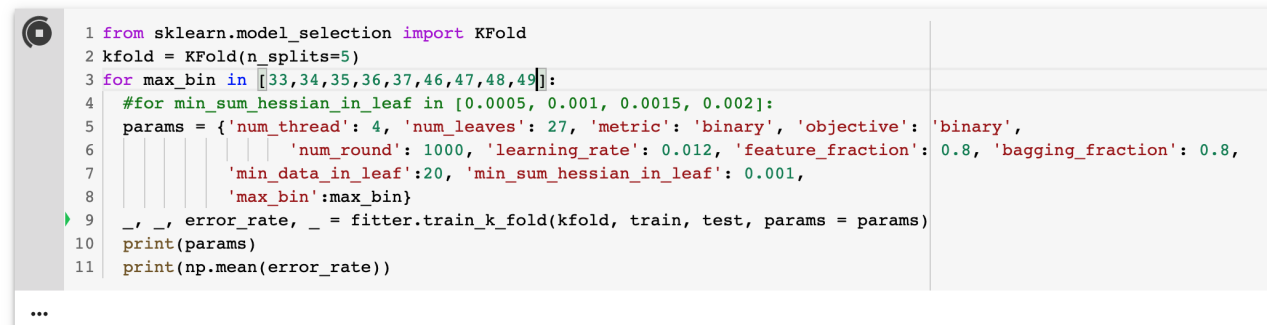
结果证明默认的通常都是不错的选择，而且发现min\_sum\_hession\_in\_leaf对结果的精度没有影响。然后单独测试max\_bin在小于默认值255时不同值表现，结果部分截图如下:



分析: 除了在参数{'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1000，'learning\_rate':0.012，'feature\_fraction':0.8，'bagging\_fraction':0.8，'min\_data\_in\_leaf':20，'min\_sum\_hessian\_in\_leaf':0.001，'max\_bin': 47}得到接近目前最优值的0.0804以外，其它的参数组合相差较大，虽然0.0804比0.08038稍大，但是根据[文档说明](https://lightgbm.readthedocs.io/en/latest/Parameters.html) [small number of bins may reduce training accuracy but may increase general power (deal with over-fitting)] 所以仍然计划测试一下周边的值，如果能找到一个参数的表现跟默认值表现一致，则更倾向于max\_bin较小时实际表现更好,继续测试max\_bin如下:



结果是令人奇怪的，在35处也收录了0.0804的误差，但是都没有255处的0.08038更好，同时也发现了比255更大的max\_bin并没有得到更好的效果，考虑到max\_bin太大的话会降低测试速度，所以不再测试更大值的范围。分析后决定进行最后一次max\_bin以观察围绕35和47附近能否得到相对最好的结果,代码如下:



然后就很神奇，当你认为不会有更好的结果准备放弃这个参数的优化时，上帝会忽然给你一个惊喜，本次测试发现27居然得到0.08022 的成绩，参数明细为:{'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1000，'learning\_rate':0.012，'feature\_fraction':0.8，'bagging\_fraction':0.8，'min\_data\_in\_leaf':20，'min\_sum\_hessian\_in\_leaf':0.001，'max\_bin': 37}，误差值为:0.08022。同时另一组参数{'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1000，'learning\_rate':0.012，'feature\_fraction':0.8，'bagging\_fraction':0.8，'min\_data\_in\_leaf':20，'min\_sum\_hessian\_in\_leaf':0.001，'max\_bin': 49}也收录了比之前更好的误差0.08034000000000004。

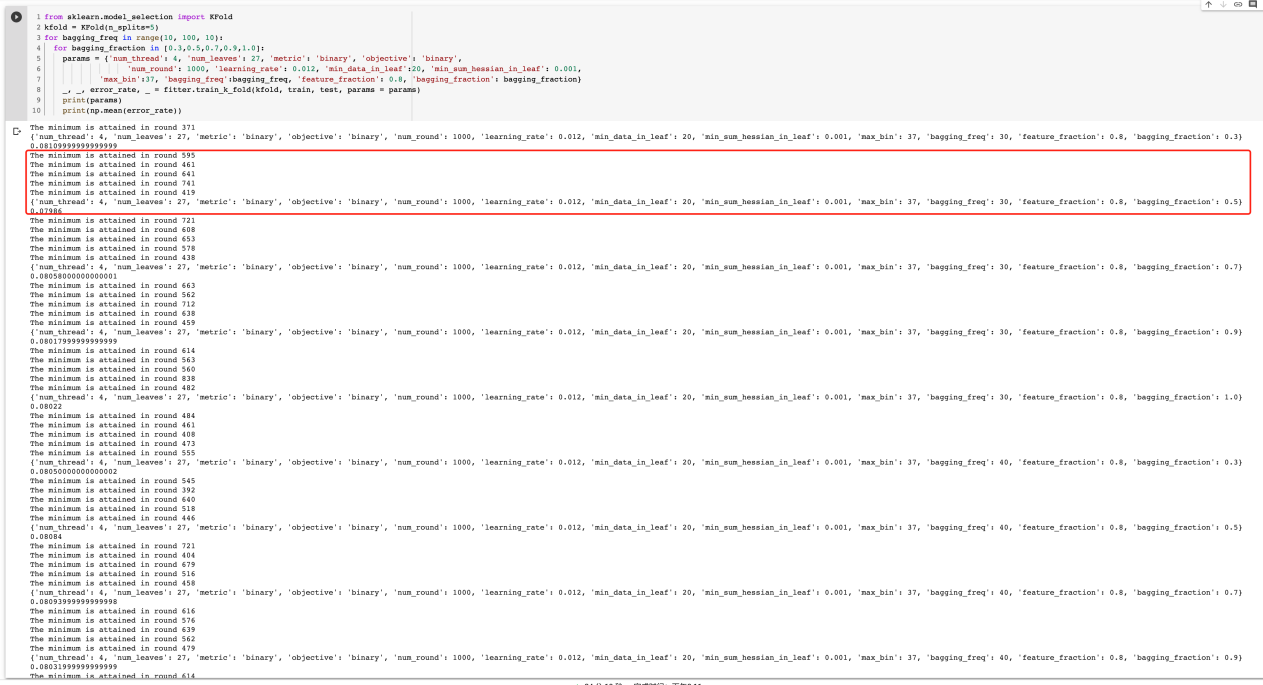
这一下来劲了呀，根据之前的结果将可能有意义的38，39都试一下如下:



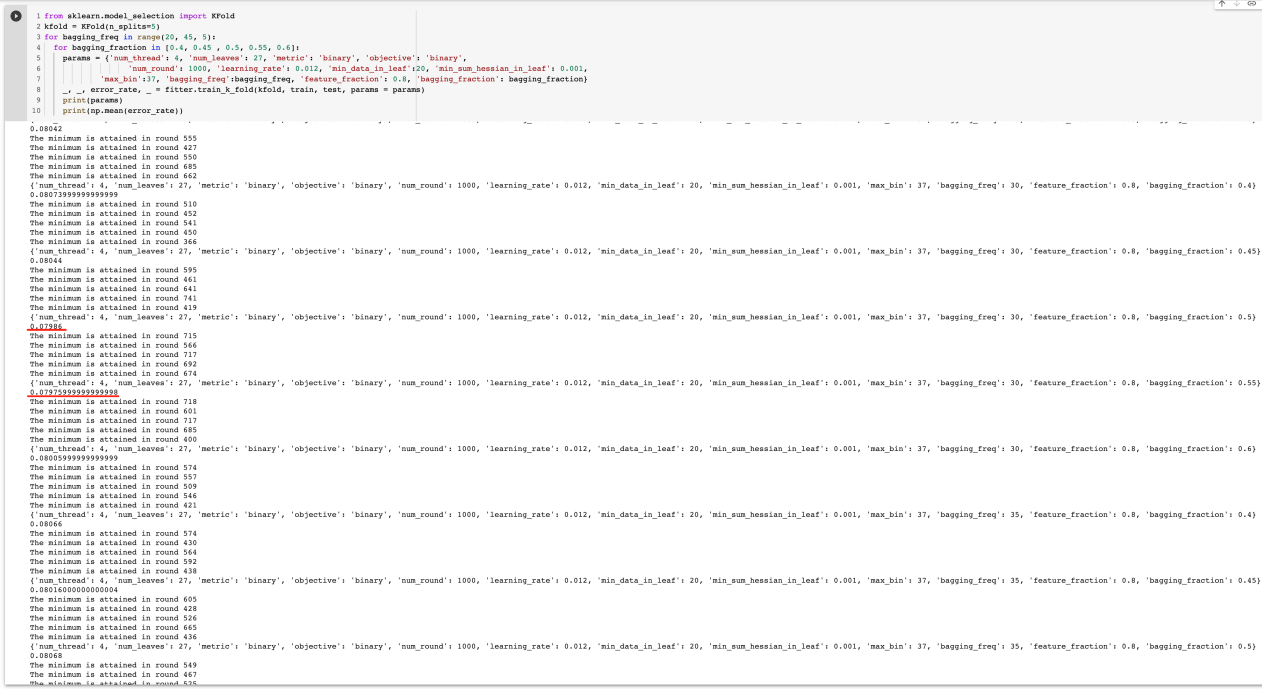
结论发现目前为止的最优解由参数组合{'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1000，'learning\_rate':0.012，'feature\_fraction':0.8，'bagging\_fraction':0.8，'min\_data\_in\_leaf':20，'min\_sum\_hessian\_in\_leaf':0.001，'max\_bin': 37}得到，误差值:0.08022，而且由于max\_bin较小在对过拟合的处理会相对好些，可以说是比默认值更乐意看到的结果了。

* 1. feature\_fraction和 bagging\_fraction

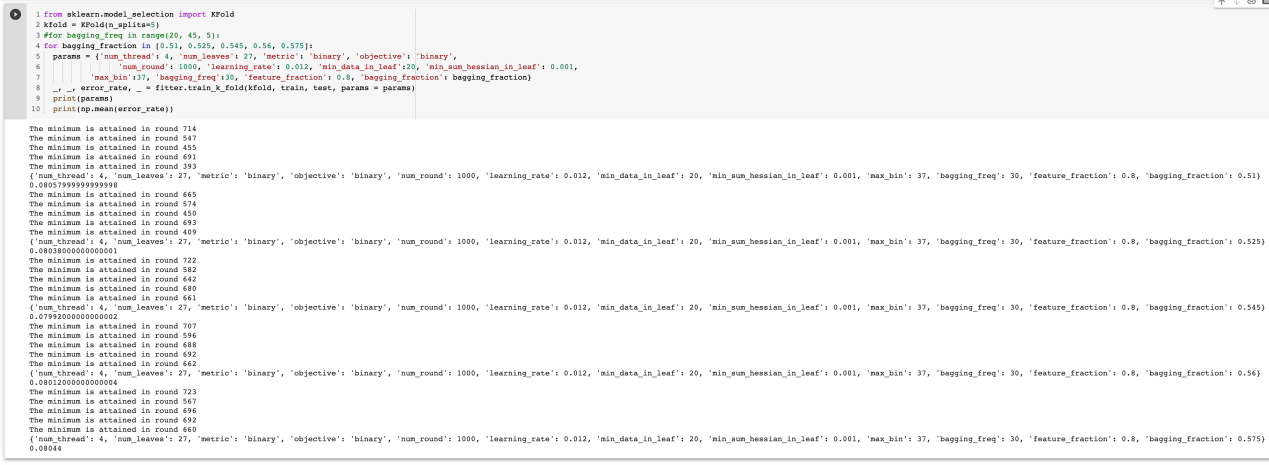
接下来归到之前的思路，继续测试参数feature\_fraction，bagging\_fraction和bagging\_freq，因为要测试bagging\_fraction的话就需要bagging\_freq大于0所以先组合看下bagging\_fraction和 bagging\_freq的情况。具体操作是取一个较大的步长范围搜索一下：bagging\_freq取10-90，bagging\_fraction则取0.3-1.0，结果如下：

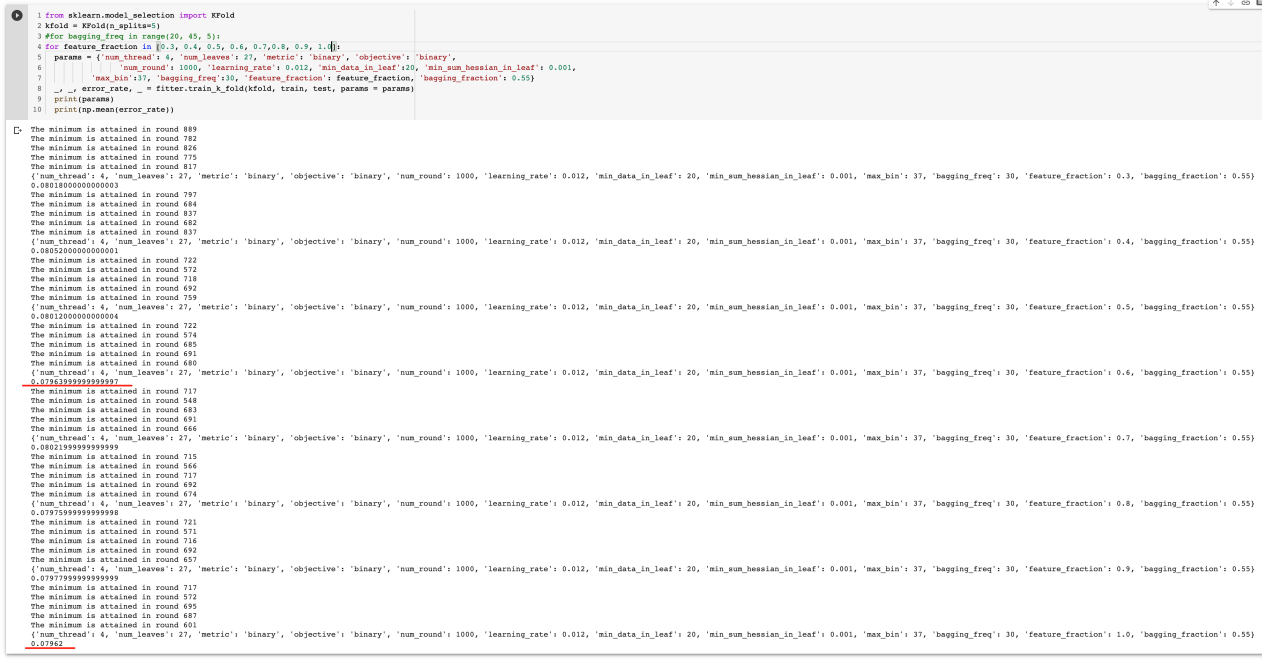


发现在参数{'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1000，'learning\_rate':0.012，'min\_data\_in\_leaf':20，'min\_sum\_hessian\_in\_leaf':0.001，'max\_bin': 37，'bagging\_freq': 30，'feature\_fraction':0.8，'bagging\_fraction':0.5}下录得了最小误差为0.07986，其余测试结果误差均大于0.08。接着在此基础上再细化步长搜索得到结果如下图：

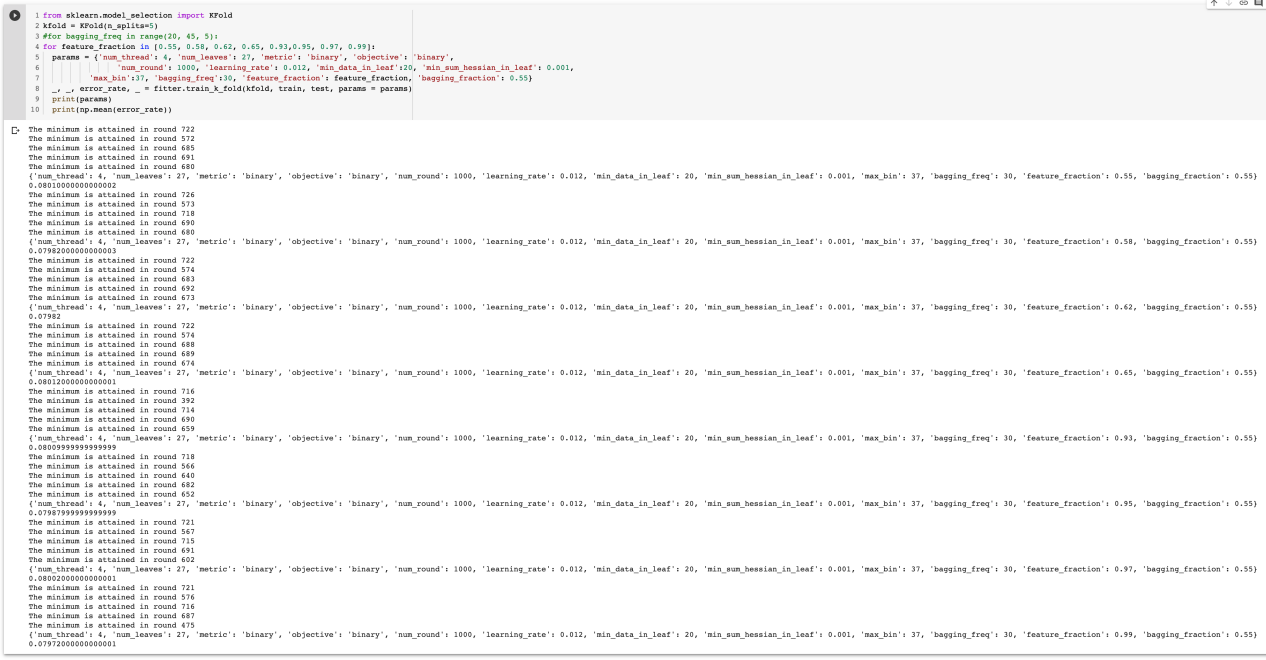


本次测试得知{'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1000，'learning\_rate':0.012，'min\_data\_in\_leaf':20，'min\_sum\_hessian\_in\_leaf':0.001，'max\_bin': 37，'bagging\_freq': 30，'feature\_fraction':0.8，'bagging\_fraction':0.55} 附近得到更小值0.07975999999999998，说明这里附近可能还有更好的选择，继续细化测试结果如下:



结果得到的最小值为0.07992反而没有上面的0.07975更好，那么认为'bagging\_freq': 30和'bagging\_fraction':0.55 是这两个参数组合最好的结果。然后针对feature\_fraction同样是进行循环测试，测试0.3到1.0之间(步长为0.1) 的表现，结果如下：

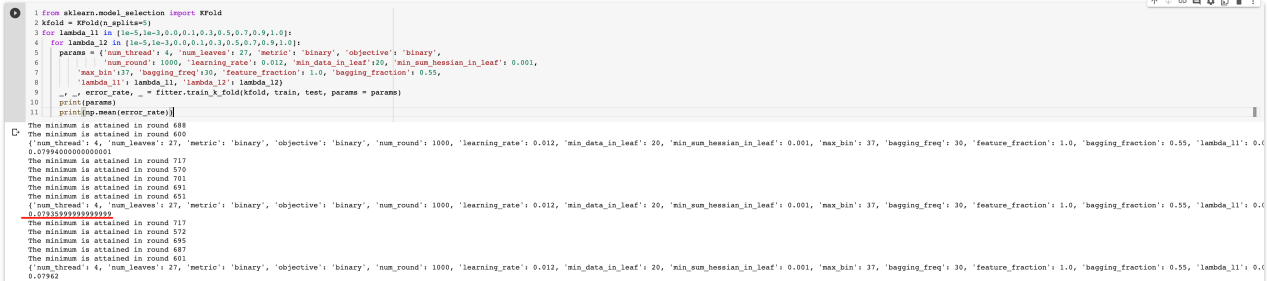
发现在0.6和1.0附近均获得了更小的误差，因为1.0已经是峰值了搜索范围不够大，所以对两个可能有更优值的范围都测试下：



结果不是很好，在这一轮搜索中没有得到更小的误差。所以 feature\_fraction优化的结果是1.0。截止目前为止最好的优化结果由参数组合{'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1000，'learning\_rate':0.012，'min\_data\_in\_leaf':20，'min\_sum\_hessian\_in\_leaf':0.001，'max\_bin': 37，'bagging\_freq': 30，'feature\_fraction':1.0，'bagging\_fraction':0.55}输出，误差为0.07962。

* 1. lambda\_l1和 lambda\_l2

因为对0.07962不是很满意，决定再试下lambda\_l1和lambda\_l2， 测试这两个参数分别在1e-5，1e-3，0.0，0.1，0.3，0.5，0.7，0.9，1.0等几个常见测试值下的组合表现



发现参数组合 {'num\_thread': 4，'num\_leaves':27，'metric':'binary'，'objective':'binary'，'num\_round':1000，'learning\_rate':0.012，'min\_data\_in\_leaf':20，'min\_sum\_hessian\_in\_leaf':0.001，'max\_bin': 37，'bagging\_freq': 30，'feature\_fraction':1.0，'bagging\_fraction':0.55，'lambda\_l1':0.0，'lambda\_l2':0.1}处获得了0.07935999999999999的误差值，到这里每次优化的幅度已经比较小了，所以测试到此为止，0.07935999999999999 成为测试最终的结果。

1. 总结和结果

本次作业过程中调参似乎总是探索性调参然后等待结果，判断参数是否有效再进行下一轮的过程。总的来说就是不停看网上经验和文档人云亦云，对自己的判断不是十分自信希望接下来有更多这方面的锻炼，同时如有错误请老师不吝指出，十分感谢。

本次作业调参在测试集上的结果为：0.07935999999999999。