结论：

训练集

0.06933999999999996

测试集

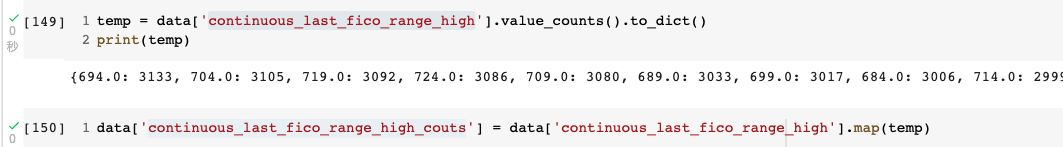
0.08362000000000003

1. 检查数据：空值、y值
2. 跑baseline（上次作业），误差：0.08006000000000002，测试集：0.0867



1. 选择模型选择变量，经过20+次观察，最终选择了

continuous\_last\_fico\_range\_high。然后进行构建变量



1. 跑分。比baseline高，有所提升，误差：0.07998000000000001，测试集：0.0.08238000000000001



1. 调参

num\_leaves:这是用于控制树模型的复杂度的主要参数。理论上设置num\_leaves = 2^max\_depth获得与树深度相同的叶子数，但实际上叶子数比树的深度更深，不受约 束容易导致过拟合，所以通常设置num\_leaves<2^max\_depth。

当num\_leaves设为[4,6,12, 31, 62, 81, 127, 256],最好的结果是

{'num\_thread': 4, 'num\_leaves': 4, 'metric': 'binary', 'objective': 'binary', 'num\_round': 4000, 'learning\_rate': 0.0075, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}

0.07951999999999997

0.08301999999999998

当learning\_rate设为[0.1, 0.08, 0.06, 0.05, 0.0075,0.0008, 0.005, 0.003]，最好的结果是:

{'num\_thread': 4, 'num\_leaves': 4, 'metric': 'binary', 'objective': 'binary', 'num\_round': 4000, 'learning\_rate': 0.0075, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8} 0.07951999999999997

0.08250000000000002

当num\_round设为[2000,3000,4000]，最好的结果是:

{'num\_thread': 4, 'num\_leaves': 4, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.0075, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8} 0.07949999999999997

0.08264000000000005

当feature\_fraction设为[1.0, 0.8, 0.7, 0.5, 0.4, 0.3]，最好的结果是:

{'num\_thread': 4, 'num\_leaves': 4, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.0075, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8} 0.07949999999999997 0.08264000000000005

当bagging\_fraction设为[1.0, 0.8, 0.75, 0.5, 0.4, 0.3]，最好的结果是，不影响:

{'num\_thread': 4, 'num\_leaves': 4, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.0075, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}

0.07949999999999997

0.08264000000000005

1. Stacking

第一层GBDT，XGB，LGB

第二层用线性回归

训练集

0.06933999999999996

测试集

0.08362000000000003

