第二次作业-橙子

测试集上的准确率为0.92016

1. 第1次调参

随机选择了参数，写for循环，先测试num\_leaves

for num\_leaves in [12, 24, 36, 60, 84, 108, 120]:

params = {'num\_thread': 4, 'num\_leaves': num\_leaves, 'metric': 'binary', 'objective': 'binary',

'num\_round': 2000, 'learning\_rate': 0.005, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}

\_, \_, error\_rate, \_ = fitter.train\_k\_fold(kfold, data\_train, data\_test, params = params)

print(params)

print(np.mean(error\_rate))

结果表现是12时最好，错误率为0.08018

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 2000, 'learning\_rate': 0.005, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}  
0.08018

{'num\_thread': 4, 'num\_leaves': 24, 'metric': 'binary', 'objective': 'binary', 'num\_round': 2000, 'learning\_rate': 0.005, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}  
0.08095999999999999

{'num\_thread': 4, 'num\_leaves': 36, 'metric': 'binary', 'objective': 'binary', 'num\_round': 2000, 'learning\_rate': 0.005, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}  
0.08102000000000002

{'num\_thread': 4, 'num\_leaves': 60, 'metric': 'binary', 'objective': 'binary', 'num\_round': 2000, 'learning\_rate': 0.005, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}  
0.08185999999999997

{'num\_thread': 4, 'num\_leaves': 84, 'metric': 'binary', 'objective': 'binary', 'num\_round': 2000, 'learning\_rate': 0.005, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}  
0.08195999999999999

{'num\_thread': 4, 'num\_leaves': 108, 'metric': 'binary', 'objective': 'binary', 'num\_round': 2000, 'learning\_rate': 0.005, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}  
0.08274

{'num\_thread': 4, 'num\_leaves': 120, 'metric': 'binary', 'objective': 'binary', 'num\_round': 2000, 'learning\_rate': 0.005, 'feature\_fraction': 0.8, 'bagging\_fraction': 0.8}  
0.08286

1. 第2次调参

调feature\_fraction

for feature\_fraction in [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]:

params = {'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary',

'num\_round': 2000, 'learning\_rate': 0.01, 'feature\_fraction': feature\_fraction, 'bagging\_fraction': 0.8}

\_, \_, error\_rate, \_ = fitter.train\_k\_fold(kfold, data\_train, data\_test, params = params)

print(params)

print(np.mean(error\_rate))

过程中发现lr有点低，在不到1000次的时候，就会停止，所以调低了lr

for feature\_fraction in [0.5, 0.6, 0.7, 0.8, 0.9]:

params = {'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary',

'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': feature\_fraction, 'bagging\_fraction': 0.8}

\_, \_, error\_rate, \_ = fitter.train\_k\_fold(kfold, data\_train, data\_test, params = params)

print(params)

print(np.mean(error\_rate))

调参结果是，feature\_fraction是0.9的时候，效果最好，error\_rate为0.08001999999999998

1. 第3次调参

调bagging\_fraction，for循环调试，但是对结果无影响，不清楚什么原因

for bagging\_fraction in [0.25, 0.5, 0.75, 1]:

params = {'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary',

'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9,

'bagging\_fraction': bagging\_fraction}

\_, \_, error\_rate, \_ = fitter.train\_k\_fold(kfold, data\_train, data\_test, params = params)

print(params)

print(np.mean(error\_rate))

1. 第4次调参

调l1参数

for lambda\_l1 in list(range(0,10)):

params = {'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary',

'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9,

'bagging\_fraction': 0.8,

'lambda\_l1': lambda\_l1}

\_, \_, error\_rate, \_ = fitter.train\_k\_fold(kfold, data\_train, data\_test, params = params)

print(params)

print(np.mean(error\_rate))

结果显示l1为1的时候最好，

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 0}  
0.08001999999999998

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 1}  
0.08000000000000003

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 2}  
0.08012

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 3}  
0.08016000000000001

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 4}  
0.08019999999999998

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 5}  
0.08046000000000002

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 6}  
0.08039999999999999

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 7}  
0.08059999999999998

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 8}  
0.08059999999999998

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 9}  
0.08053999999999999

lamaba\_l1等于1的时候效果最好

1. 第5次调参

在l1的基础上，直接添加l2参数测试

for lambda\_l2 in list(range(0,7)):

params = {'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary',

'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9,

'bagging\_fraction': 0.8,

'lambda\_l1': 1,

'lambda\_l2': lambda\_l2}

\_, \_, error\_rate, \_ = fitter.train\_k\_fold(kfold, data\_train, data\_test, params = params)

print(params)

print(np.mean(error\_rate))

结果显示l1为1，l2为4的组合，效果最好0.07996

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 1, 'lambda\_l2': 0}  
0.08000000000000003

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 1, 'lambda\_l2': 1}  
0.08015999999999998

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 1, 'lambda\_l2': 2}  
0.08028000000000002

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 1, 'lambda\_l2': 3}  
0.08012

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 1, 'lambda\_l2': 4}  
0.07996

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 1, 'lambda\_l2': 5}  
0.07998000000000001

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l1': 1, 'lambda\_l2': 6}  
0.08015999999999998

lambda\_l2为4的时候，表现较好

1. 第6次调参

不添加l1参数，只使用l2

for lambda\_l2 in list(range(0,10)):

params = {'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary',

'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9,

'bagging\_fraction': 0.8,

'lambda\_l2': lambda\_l2}

\_, \_, error\_rate, \_ = fitter.train\_k\_fold(kfold, data\_train, data\_test, params = params)

print(params)

print(np.mean(error\_rate))

结果显示l2的时候，结果较好，好于之前

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 0}  
0.08001999999999998

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 1}  
0.08009999999999999

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 2}  
0.07984

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 3}  
0.08030000000000001

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 4}  
0.08009999999999999

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 5}  
0.08008000000000001

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 6}  
0.08000000000000003

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 7}  
0.07998000000000001

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 8}  
0.08005999999999999

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 9}  
0.0799

1. 第7次调参

额外测试了l2在10-20的情况，结果不好

1. 第8次调参

调试main\_gain\_to\_split

for min\_gain\_to\_split in [0.0, 0.24, 0.48, 0.72, 0.96, 1.0]:

params = {'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary',

'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9,

'bagging\_fraction': 0.8,

'lambda\_l2': 2,

'min\_gain\_to\_split': min\_gain\_to\_split}

\_, \_, error\_rate, \_ = fitter.train\_k\_fold(kfold, data\_train, data\_test, params = params)

print(params)

print(np.mean(error\_rate))

参数为0的时候，效果最好

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 2, 'min\_gain\_to\_split': 0.0}  
0.07984

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 2, 'min\_gain\_to\_split': 0.24}  
0.08005999999999999

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 2, 'min\_gain\_to\_split': 0.48}  
0.07988000000000002

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 2, 'min\_gain\_to\_split': 0.72}  
0.07996

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 2, 'min\_gain\_to\_split': 0.96}  
0.08005999999999999

{'num\_thread': 4, 'num\_leaves': 12, 'metric': 'binary', 'objective': 'binary', 'num\_round': 3000, 'learning\_rate': 0.005, 'feature\_fraction': 0.9, 'bagging\_fraction': 0.8, 'lambda\_l2': 2, 'min\_gain\_to\_split': 1.0}  
0.0799

'min\_gain\_to\_split': 0.0的表现最好