1,11,21....201

{'n\_estimators': 201}----0.9817533330914605

param\_grid = {

'n\_estimators': np.arange(1,211,10),

}

regressor=LGBMRegressor(

learning\_rate=0.1,

random\_state=90)

GS=GridSearchCV(regressor,param\_grid,cv=k)

GS.fit(x\_train,y\_train)

print(GS.best\_params\_)

print(GS.best\_score\_)

3...20

#{'max\_depth': 18}----0.9818521630436967

param\_grid={

'max\_depth': np.arange(3,21,1),

}

regressor=LGBMRegressor(n\_estimators=201,

learning\_rate=0.1,

random\_state=90)

GS=GridSearchCV(regressor,param\_grid,cv=k)

GS.fit(x\_train,y\_train)

print(GS.best\_params\_)

print(GS.best\_score\_)

1,11,21...101

{'num\_leaves': 91}-----0.9872405075721262

param\_grid={

'num\_leaves': np.arange(1,111,10),

}

regressor=LGBMRegressor(n\_estimators=201,

learning\_rate=0.1,

max\_depth=18,

random\_state=90)

GS=GridSearchCV(regressor,param\_grid,cv=k)

GS.fit(x\_train,y\_train)

print(GS.best\_params\_)

print(GS.best\_score\_)

1,11,21...101

#{'min\_data\_in\_leaf': 1}---0.988489817505646

param\_grid={

'min\_data\_in\_leaf': np.arange(1,111,10),

}

regressor=LGBMRegressor(n\_estimators=201,

learning\_rate=0.1,

max\_depth=18,

num\_leaves=91,

random\_state=90)

GS=GridSearchCV(regressor,param\_grid,cv=k)

GS.fit(x\_train,y\_train)

print(GS.best\_params\_)

print(GS.best\_score\_)

5,10,15...255

#{'max\_bin': 90}---0.9889003050112304

param\_grid={

'max\_bin': np.arange(5,256,5),

}

regressor=LGBMRegressor(n\_estimators=201,

learning\_rate=0.1,

max\_depth=18,

num\_leaves=91,

min\_data\_in\_leaf=1,

random\_state=90)

GS=GridSearchCV(regressor,param\_grid,cv=k)

GS.fit(x\_train,y\_train)

print(GS.best\_params\_)

print(GS.best\_score\_)

0,0.1...1

#{'feature\_fraction': 0.5}---0.990569601079493

param\_grid={

'feature\_fraction': np.linspace(0,1,11),

}

regressor=LGBMRegressor(n\_estimators=201,

learning\_rate=0.1,

max\_depth=18,

num\_leaves=91,

min\_data\_in\_leaf=1,

max\_bin=90,

random\_state=90)

GS=GridSearchCV(regressor,param\_grid,cv=k)

GS.fit(x\_train,y\_train)

print(GS.best\_params\_)

print(GS.best\_score\_)

0,0.1...1

#{'bagging\_fraction': 0.1}---0.990569601079493

param\_grid={

'bagging\_fraction': np.linspace(0,1,11),

}

regressor=LGBMRegressor(n\_estimators=201,

learning\_rate=0.1,

max\_depth=18,

num\_leaves=91,

min\_data\_in\_leaf=1,

max\_bin=90,

feature\_fraction=0.5,

random\_state=90)

GS=GridSearchCV(regressor,param\_grid,cv=k)

GS.fit(x\_train,y\_train)

print(GS.best\_params\_)

print(GS.best\_score\_)

0,10,20,...100

{'bagging\_freq': 0}----0.990569601079493

np.arange(0,101,10),

regressor=LGBMRegressor(n\_estimators=201,

learning\_rate=0.1,

max\_depth=18,

num\_leaves=91,

min\_data\_in\_leaf=1,

max\_bin=90,

feature\_fraction=0.5,

bagging\_fraction=0.1,

random\_state=90)

GS=GridSearchCV(regressor,param\_grid,cv=k)

GS.fit(x\_train,y\_train)

print(GS.best\_params\_)

print(GS.best\_score\_)

0,0.05,0.1...1

#{'reg\_alpha': 0.25}----0.9908192355263905

np.linspace(0,1,21),

0,0.05,0.1...1

#{'reg\_lambda': 0.0}---0.9908192355263905

np.linspace(0,1,21),

#{'min\_split\_gain': 0.0}---0.9908192355263905

np.linspace(0,1,21),

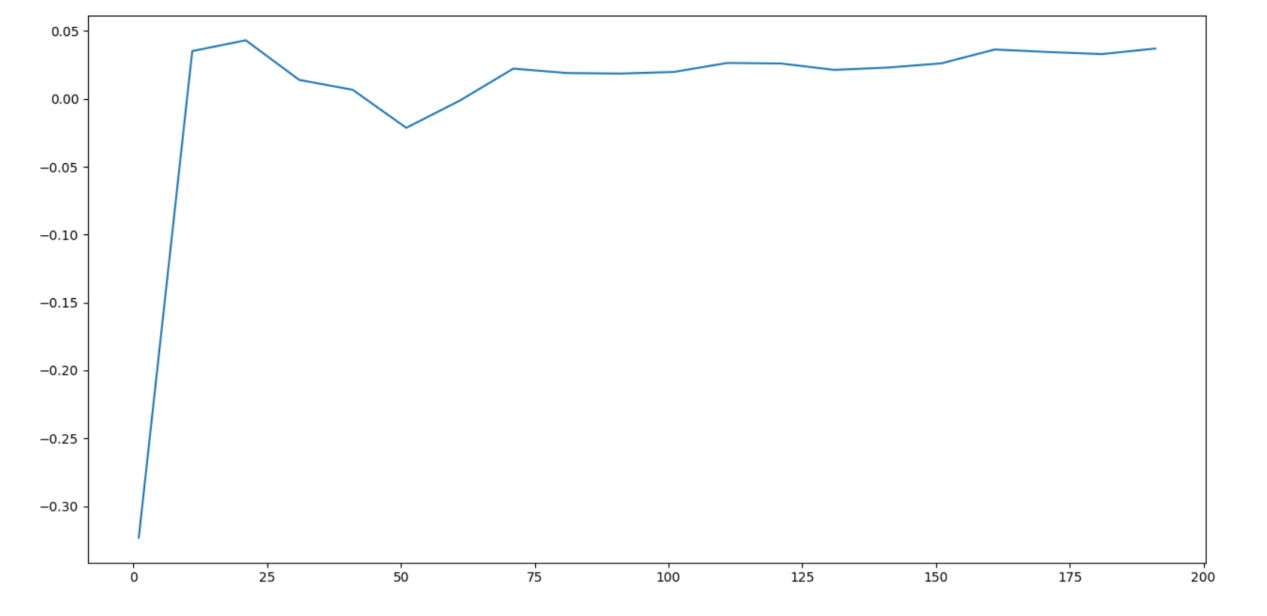
#{'learning\_rate': 0.45}----0.9800589612047853

param\_grid={

'learning\_rate': np.linspace(0,1,21),

}

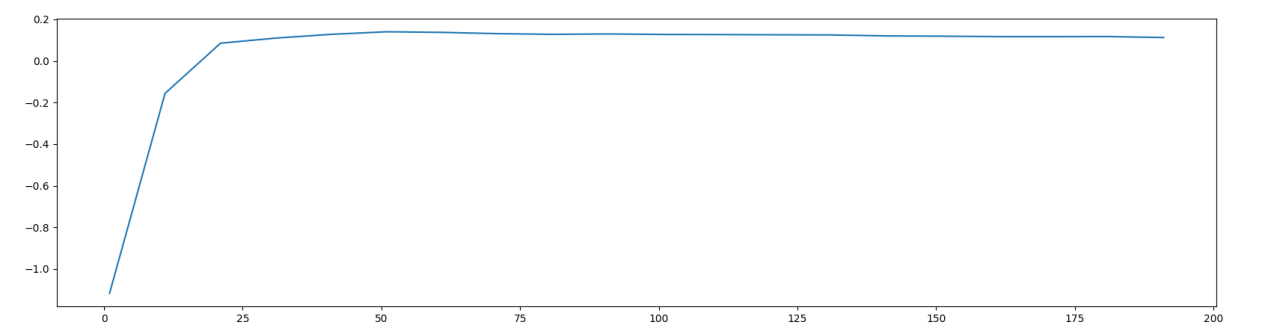
RF.。。。。----明天完整调参以下 看看最终情况



.....

这调个啥？---换个模型？-----也许深度学习bp?

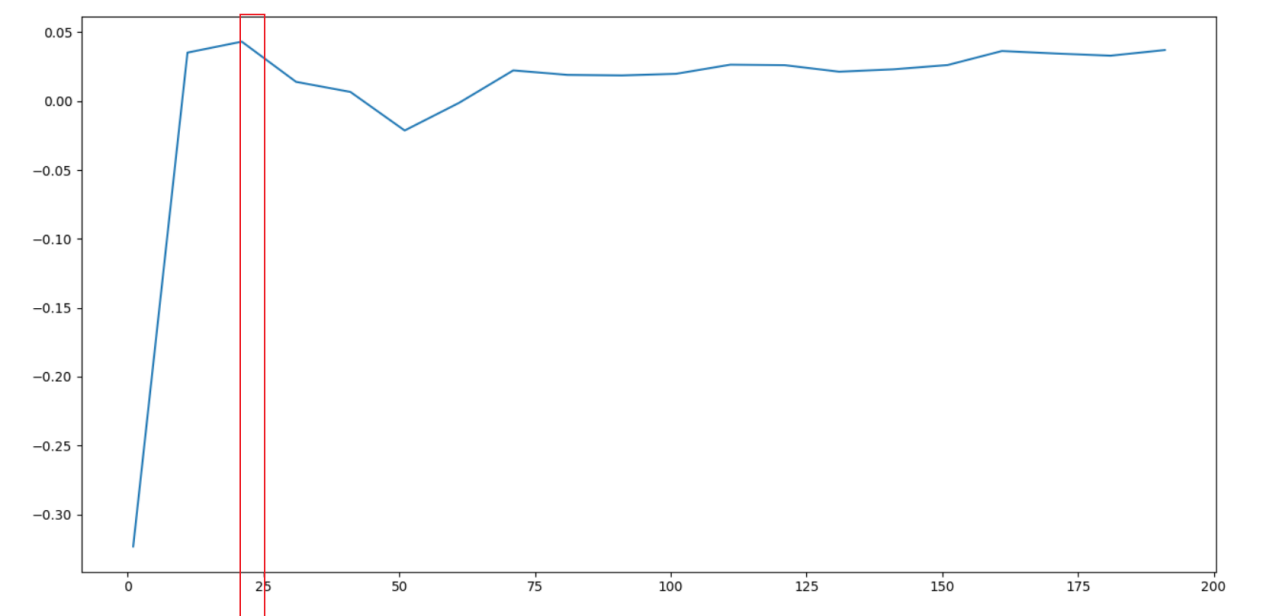
LGBM。。。。



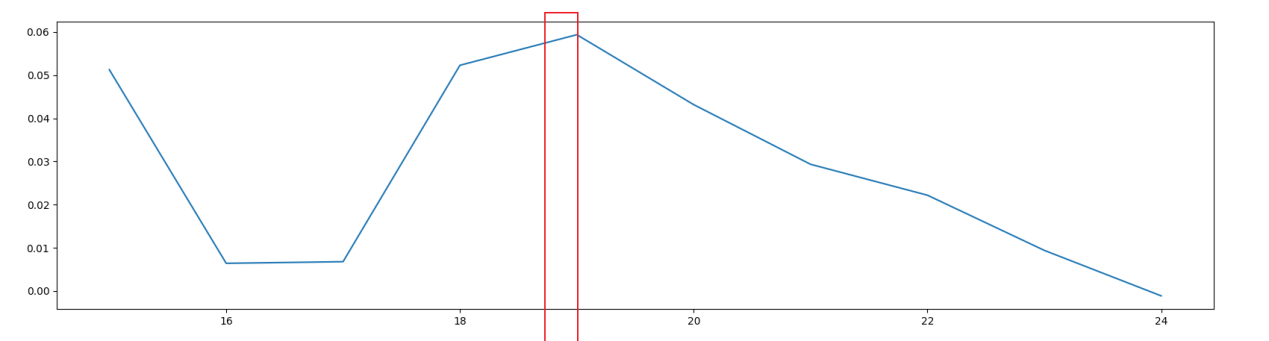
那我来个主成分分析？

#n\_estimators

#0.04318035905162898 21............这。。。。



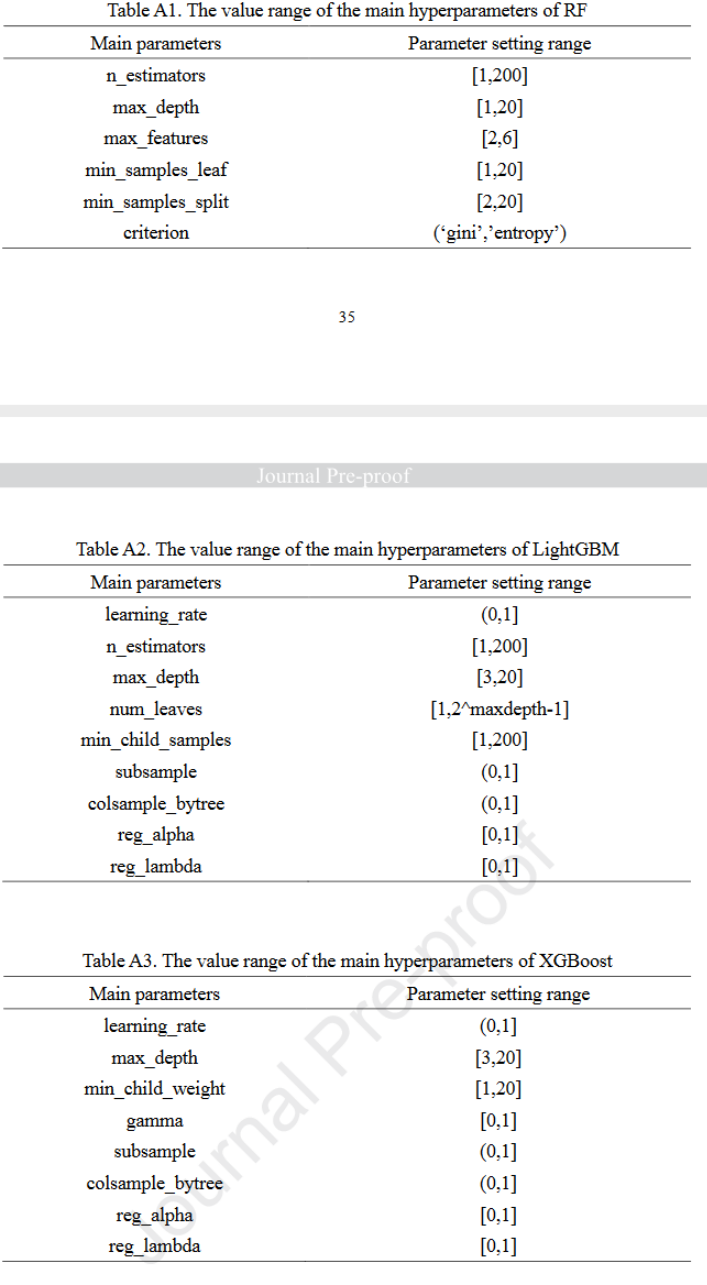
#0.05938686370704702 4----19



'max\_depth': np.arange(1,20,1), # 调整树的深度

#{'max\_depth': 19}----0.9841352601456231---没看错吧。。。

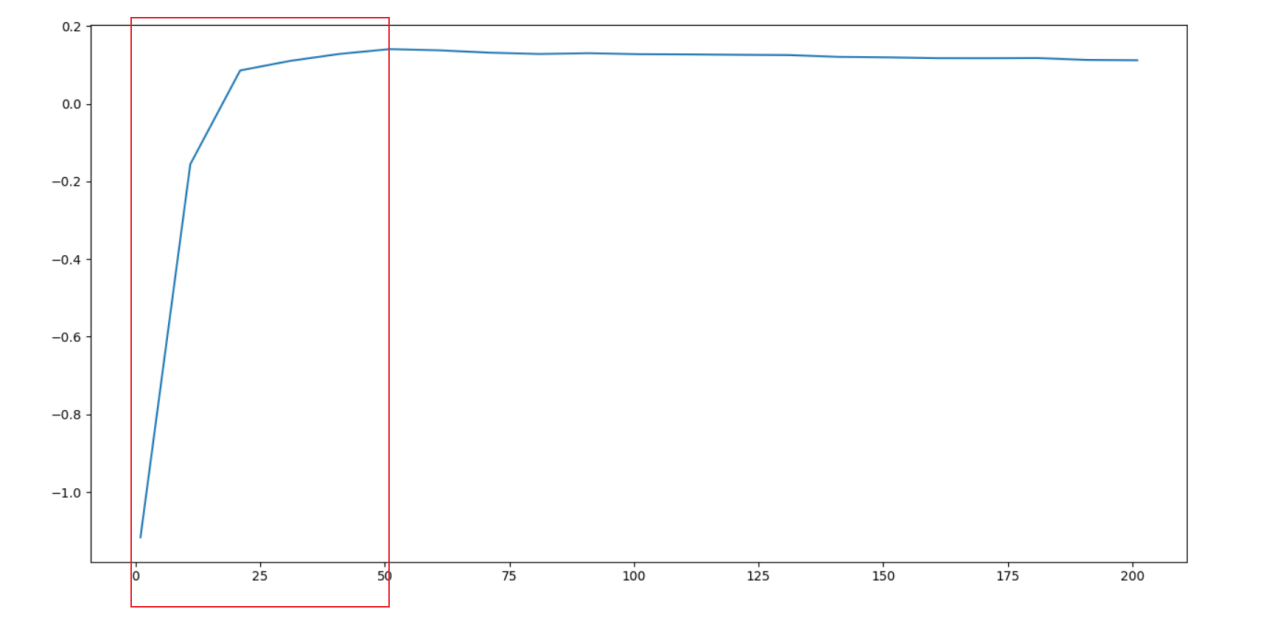
按这个表格来吧

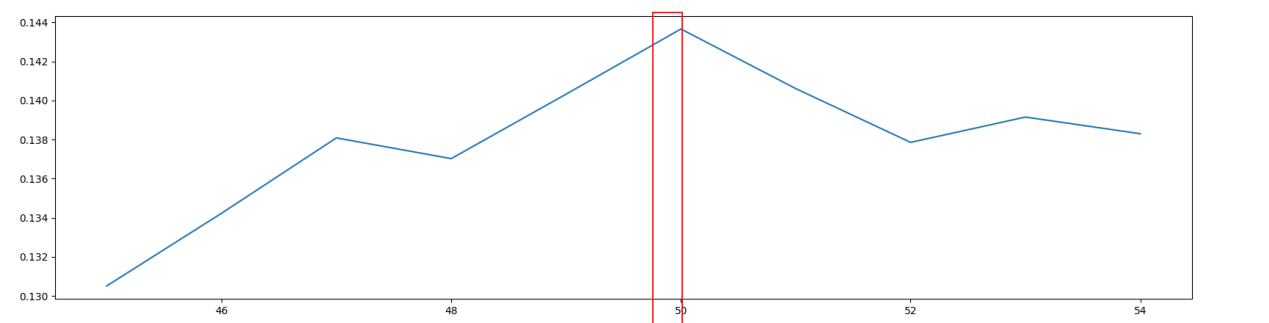


#n\_estimators

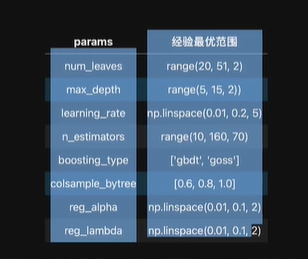
1-200

#0.1406059435903892 51........这。。。。

#0.14365427927149074 5---50



<https://www.bilibili.com/video/BV13h4y147GK?p=13&vd_source=a6f120aa01d7d9a8c8667419efef8181>

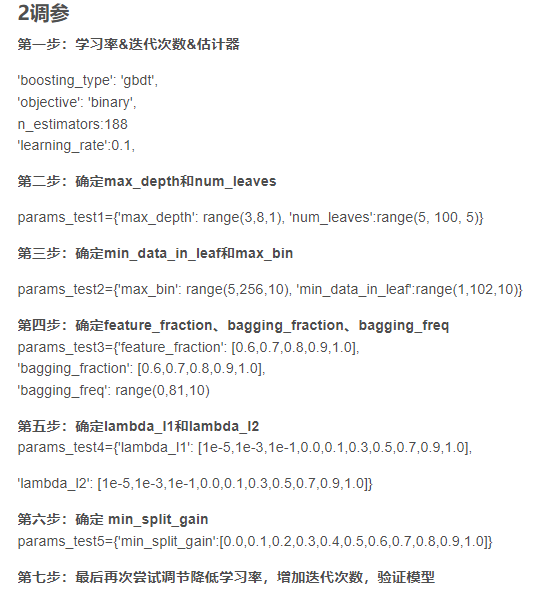


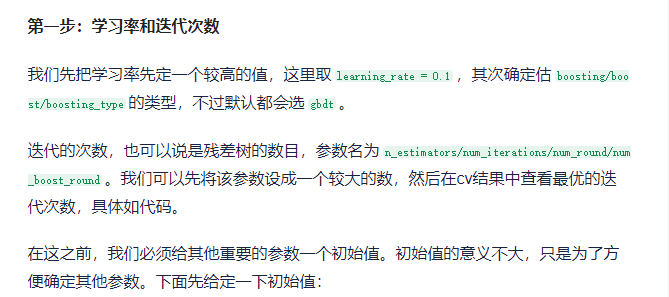
<https://blog.csdn.net/qq_35679701/article/details/107239487>

<https://blog.51cto.com/u_15274944/5042180>

<https://www.jianshu.com/p/3f114699c6ed>

[参考这个步骤，范围再说](https://www.jianshu.com/p/3f114699c6ed)

[](https://www.jianshu.com/p/3f114699c6ed)

[](https://www.jianshu.com/p/3f114699c6ed)



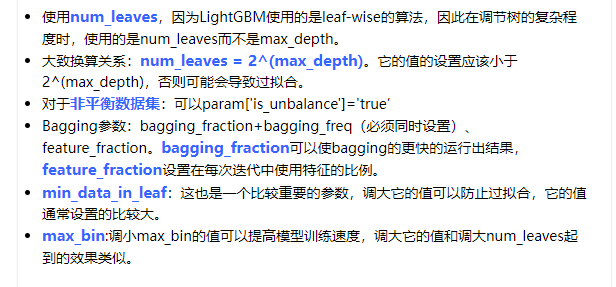
3-10

#{'max\_depth': 10}---0.9559120010637996

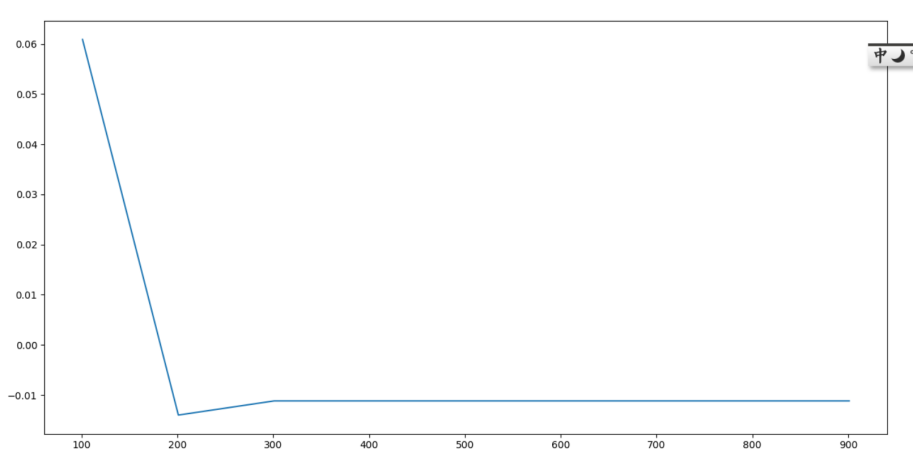
param\_grid={

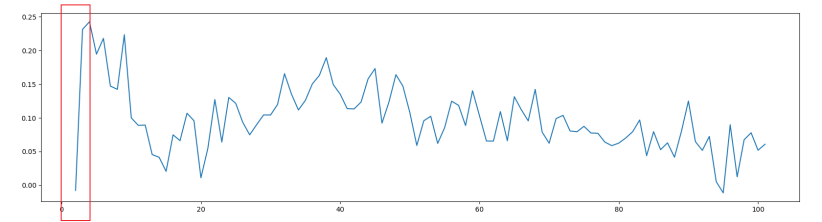
'max\_depth': np.arange(3,11,1),

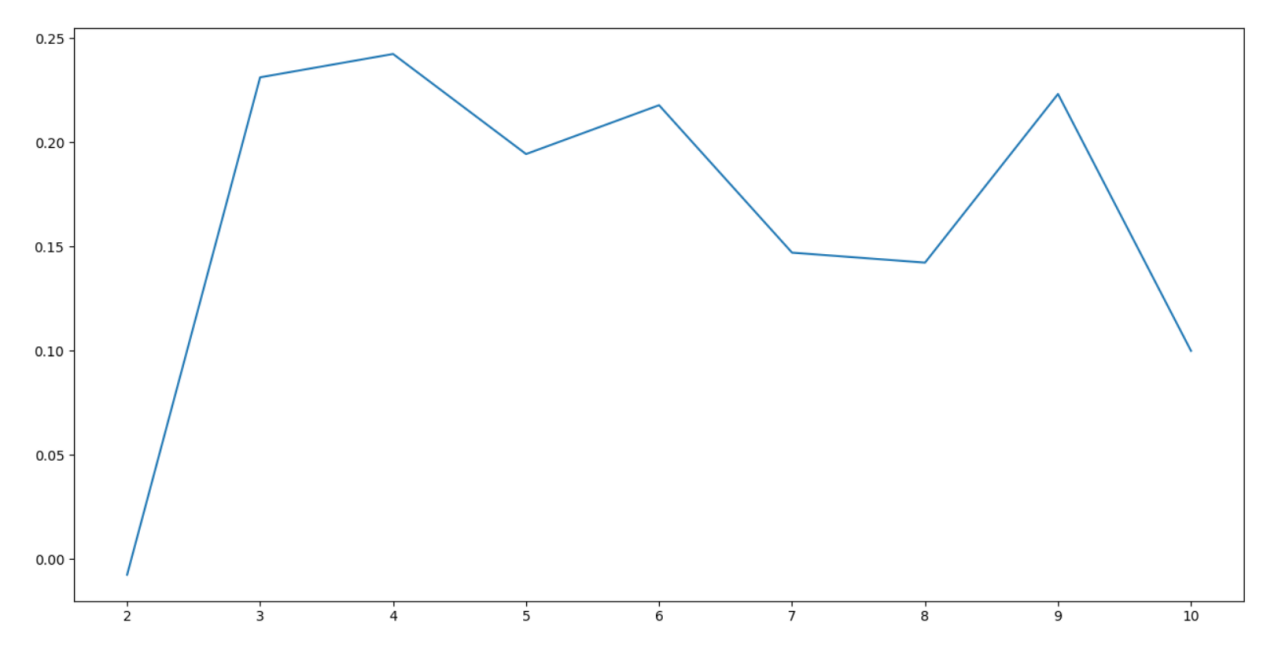
}



1-2^maxDepth-1







#num\_leaves------4

1-100

#{'min\_data\_in\_leaf': 5}------0.9586173274474092--num\_leaves=31---

#{'min\_data\_in\_leaf': 56}-----0.8295643474004241

param\_grid={

'min\_data\_in\_leaf': np.arange(1,101,1),

}

5-255

#{'max\_bin': 255}---0.9586173274474092--默认

param\_grid={

'max\_bin': np.arange(5,256,5),

}

0-1

#{'feature\_fraction': 0.6}----0.9595528477936286

param\_grid={

'feature\_fraction': np.linspace(0.1,1,10),

}

0-1

#{'bagging\_fraction': 0.1}---0.9595528477936286

param\_grid={

'bagging\_fraction': np.linspace(0.1,1,10),

}

0-100

#{'bagging\_freq': 0}---0.9595528477936286

param\_grid={

'bagging\_freq': np.arange(0,101,1),

}

0-1

#{'reg\_alpha': 0.30000000000000004}----0.9598494540684666

param\_grid={

'reg\_alpha': np.linspace(0,1,10),

}

0-1

#{'reg\_lambda': 0.0}---0.9598494540684666--默认

param\_grid={

'reg\_lambda': np.linspace(0,1,11),

}

是的，`lambda\_l1` 和 `lambda\_l2` 在 LightGBM 中分别对应于正则化的参数，而 `reg\_alpha` 和 `reg\_lambda` 是与之等效的参数。它们的作用相同，只是命名不同。

具体来说：

- `lambda\_l1` 和 `reg\_alpha`：都是 L1 正则化的强度参数。

- `lambda\_l2` 和 `reg\_lambda`：都是 L2 正则化的强度参数。

它们的取值范围是非负实数，通常在 [0, +∞) 之间。默认情况下，它们的取值都是 0，表示不应用正则化。你可以根据实际问题的需要调整这些参数，以控制模型的正则化强度。使用交叉验证等方法可以帮助你找到最佳的参数取值，以优化模型性能。

#{'min\_split\_gain': 0.0}---0.9598494540684666--默认

param\_grid={

'min\_split\_gain': np.linspace(0,1,11),

}

#{'learning\_rate': 0.45}----0.9800589612047853

param\_grid={

'learning\_rate': np.linspace(0,1,21),

}