验证集---信息泄露？---交叉验证调参x、y

<https://blog.csdn.net/m0_61399808/article/details/132136754>



https://blog.csdn.net/wzk4869/article/details/126256780?utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~baidujs\_baidulandingword~default-1-126256780-blog-107289061.235^v40^pc\_relevant\_3m\_sort\_dl\_base3&spm=1001.2101.3001.4242.2&utm\_relevant\_index=4



------------------------11/26-------------------

1,2,3...20

#{'max\_depth': 6}---0.7930430112892463

param\_grid = {

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

}

k=5

regressor=DecisionTreeRegressor(

random\_state=90)

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

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

print(GS.best\_params\_)

print(GS.best\_score\_)

#{'max\_features': 'auto'}-----0.7930430112892463---不变---默认取值

param\_grid = {

'max\_features': ['auto','sqrt','log2',None]

}

max\_features的默认最小值是sqrt(n\_features)，

regressor=DecisionTreeRegressor(

max\_depth=6,

random\_state=90)

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

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

print(GS.best\_params\_)

print(GS.best\_score\_)

1,2,3....20

#{'min\_samples\_leaf': 1}-----0.7930430112892463---不变---默认取值

param\_grid = {

'min\_samples\_leaf': np.arange(1,21,1),

}

regressor=DecisionTreeRegressor(

max\_depth=6,

max\_features='auto',

random\_state=90)

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

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

print(GS.best\_params\_)

print(GS.best\_score\_)

2,3,4,...20

#{'min\_samples\_split': 4}---0.8065468766626326

param\_grid = {

'min\_samples\_split': np.arange(2,21,1),

}

regressor=DecisionTreeRegressor(

max\_depth=6,

max\_features='auto',

min\_samples\_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\_)

MAE\_train: 0.000826185119047619

MSE\_train: 1.9723919157291666e-06

RMSE\_train: 0.0014044187109723248

MAPE\_train: 0.42020514886043464

r2\_score\_train: 0.9642723328696964

EV\_train: 0.9642723328696964

-------------------

MAE\_test: 0.0011232186904761906

MSE\_test: 2.6938371391962314e-06

RMSE\_test: 0.0016412913023580644

MAPE\_test: 0.4103640137895974

r2\_score\_test: 0.9445934257846557

EV\_test: 0.9454912863407611

----------------------------------

MAE\_train: 0.0006059663865546221

MSE\_train: 7.840849525694792e-07

RMSE\_train: 0.000885485715621364

MAPE\_train: 1.0000675342942185

r2\_score\_train: 0.9857971805887644

EV\_train: 0.9859607735640099

-------------------

MAE\_test: 0.001021136512605042

MSE\_test: 1.7458930922491596e-06

RMSE\_test: 0.00132132247852262

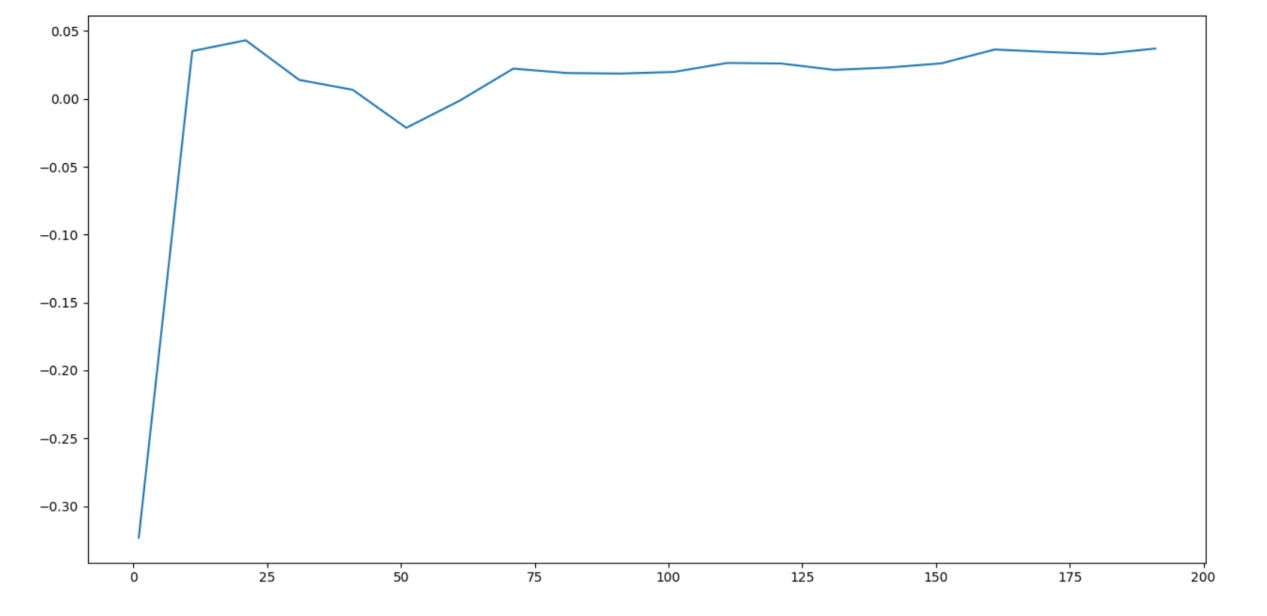
MAPE\_test: 2.65127089609236

r2\_score\_test: 0.9640906446123826

EV\_test: 0.9662024268206604

-------------------

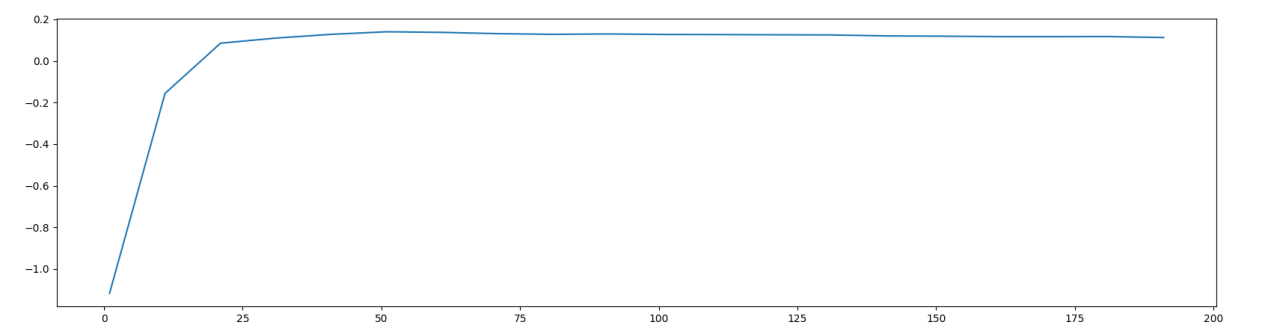
RF.。。。。----明天完整调参以下 看看最终情况



.....

这调个啥？---换个模型？-----也许深度学习bp?

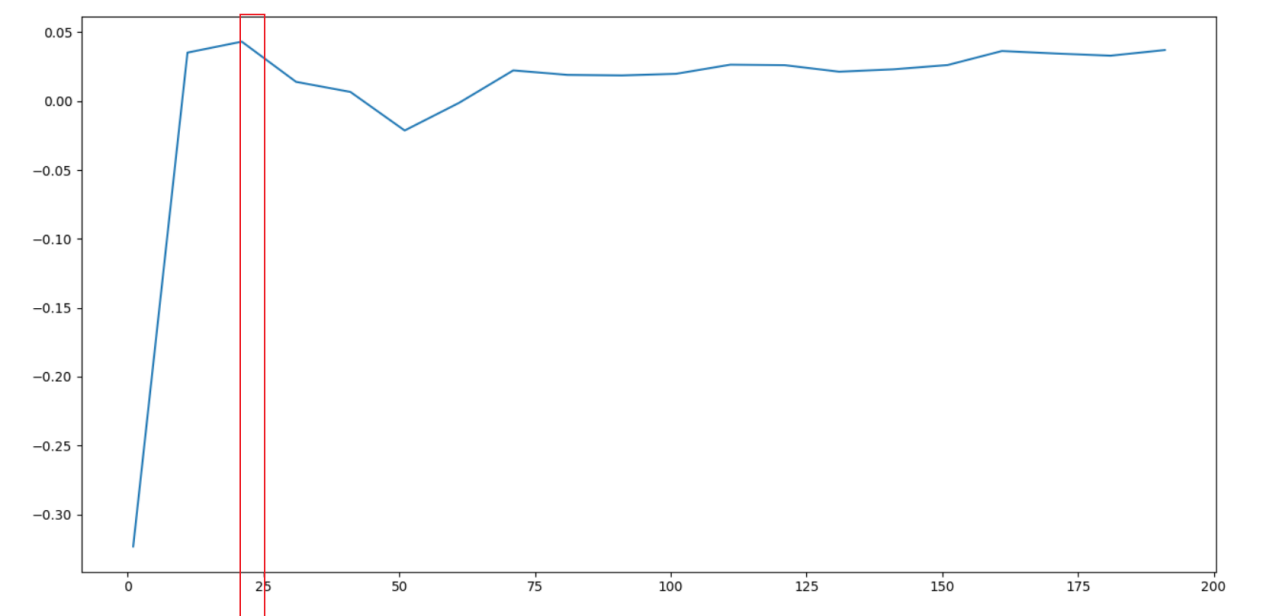
LGM。。。。



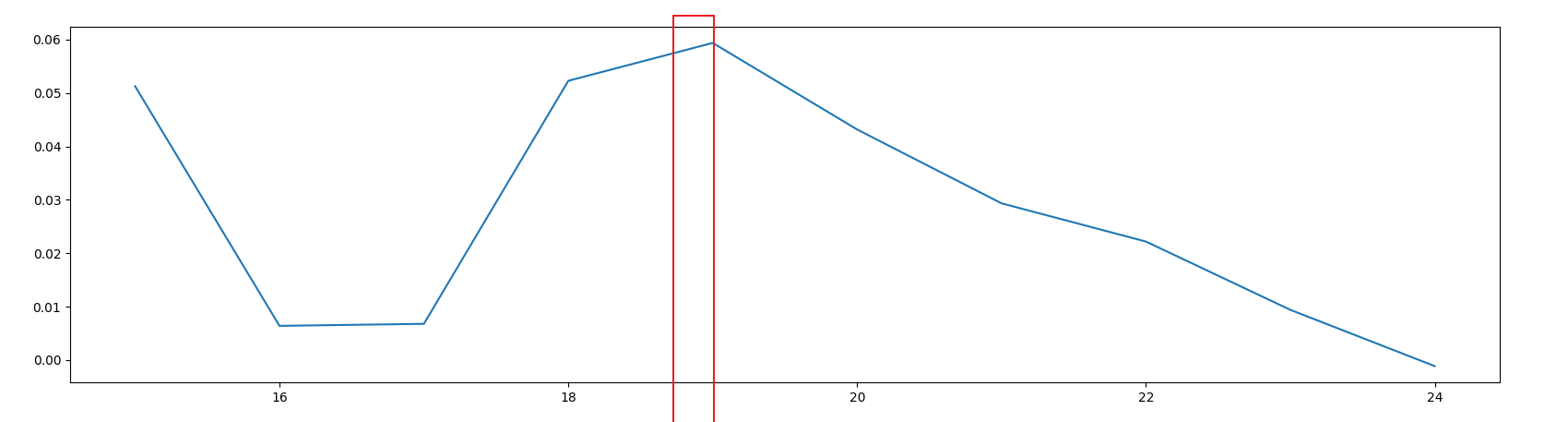
那我来个主成分分析？

#n\_estimators

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

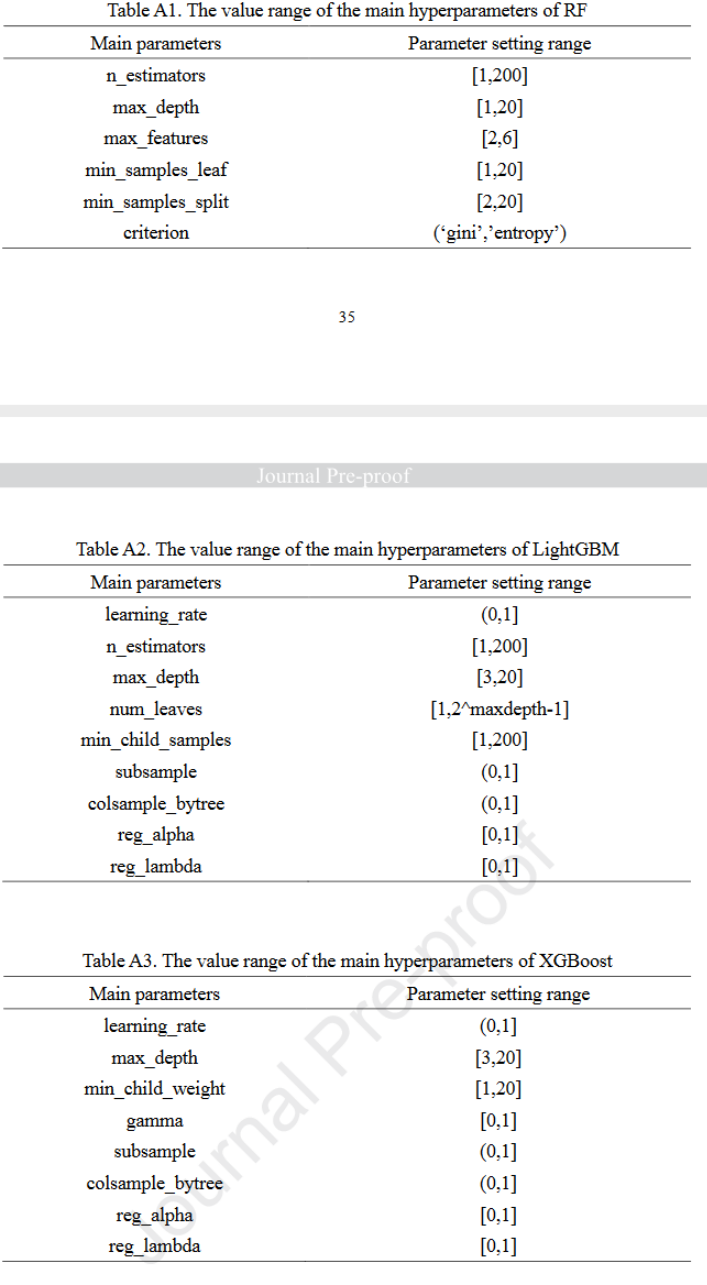


#0.05938686370704702 4----19

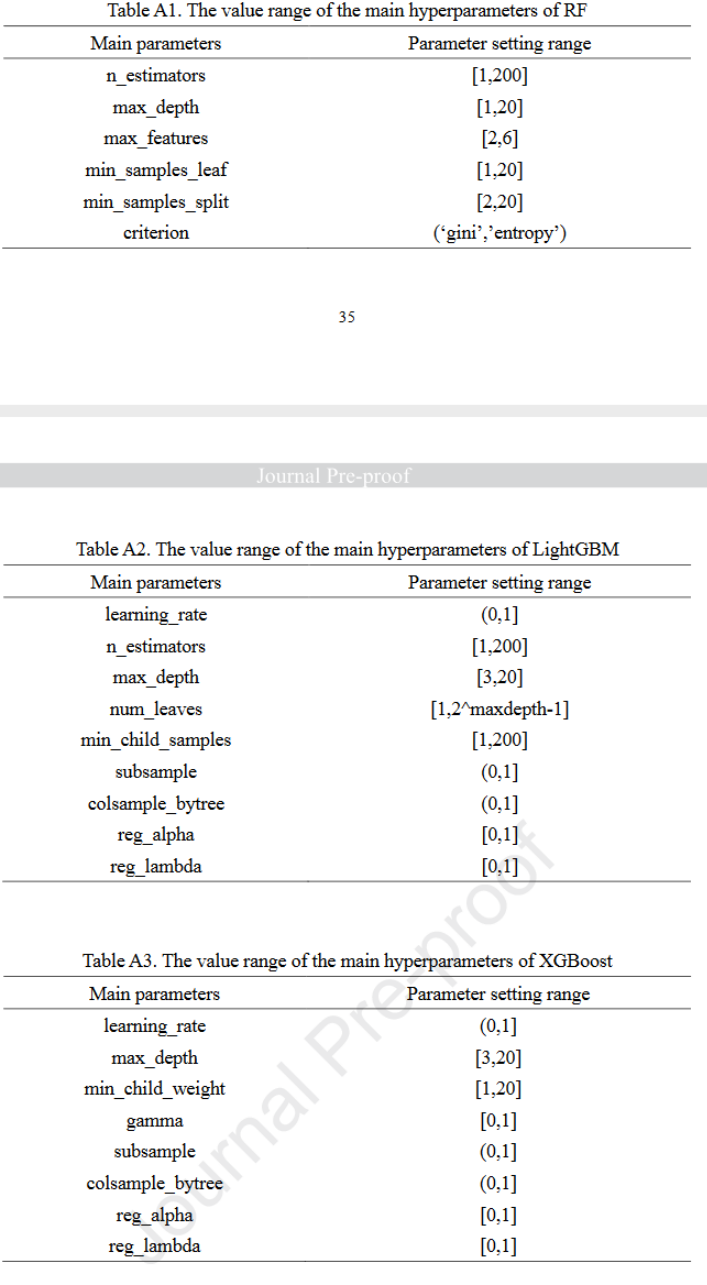


'max\_depth': np.arange(1,20,1), # 调整树的深度

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



按这个表格来吧



np.arange(1,20,1)什么意思，举例说明

`np.arange(1, 20, 1)` 是NumPy库中的一个函数调用，用于创建一个数组，其元素为从1开始、步长为1、不包括20的整数序列。具体来说，这个函数的参数如下：

- `start`: 序列的起始值，这里是1。

- `stop`: 序列的结束值，但不包括该值，这里是20。

- `step`: 序列中每个元素之间的步长，这里是1。

让我们通过一个示例来说明：

```python

import numpy as np

# 使用np.arange创建一个数组

arr = np.arange(1, 20, 1)

# 输出数组

print(arr)

```

输出：

```

[ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]

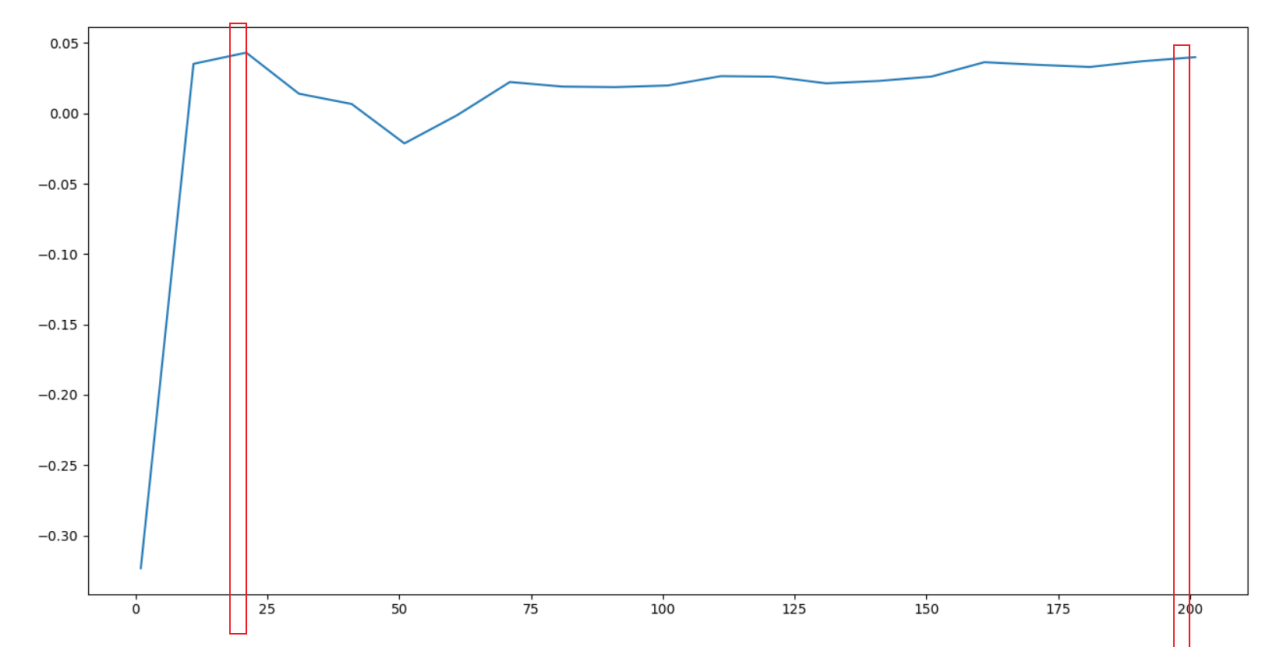
```

在这个例子中，`np.arange(1, 20, 1)` 创建了一个包含整数的NumPy数组，从1开始，以1为步长，一直到19（不包括20）。这是一个方便的函数，用于生成等差数列。你可以根据需要调整起始值、结束值和步长。

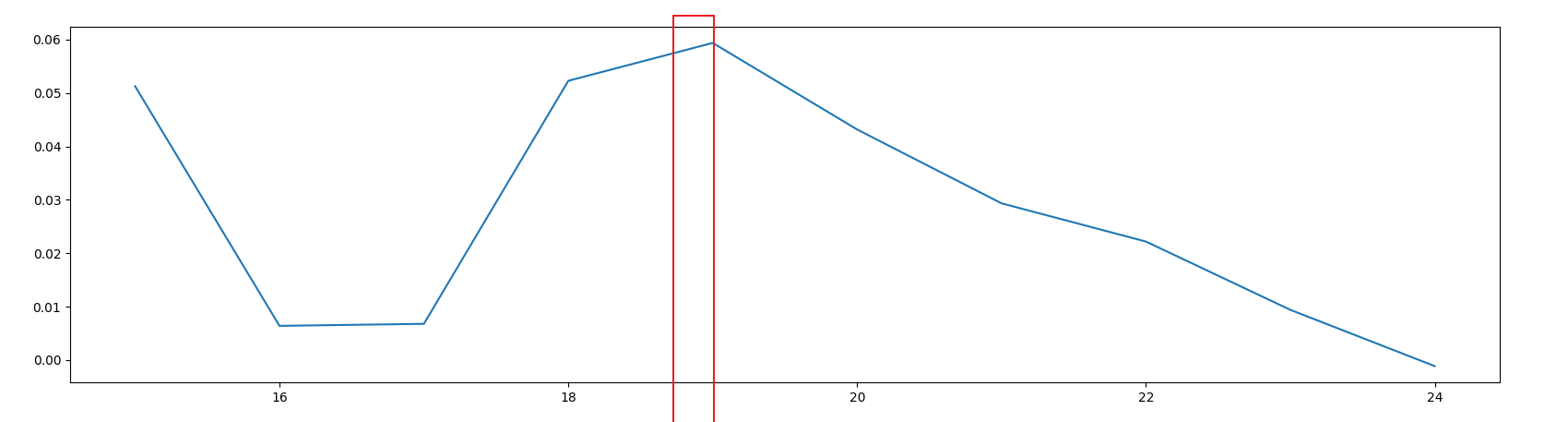
#n\_estimators

1-200

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



#0.05938686370704702 4----19



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

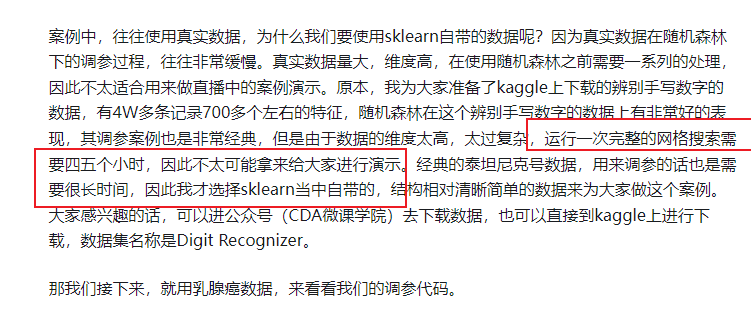
#{'max\_depth': 19}----0.9841352601456231---没看错吧。。。

param\_grid = {

'max\_depth': np.arange(1,21,1), # 调整树的深度

}

<https://zhuanlan.zhihu.com/p/59017155>





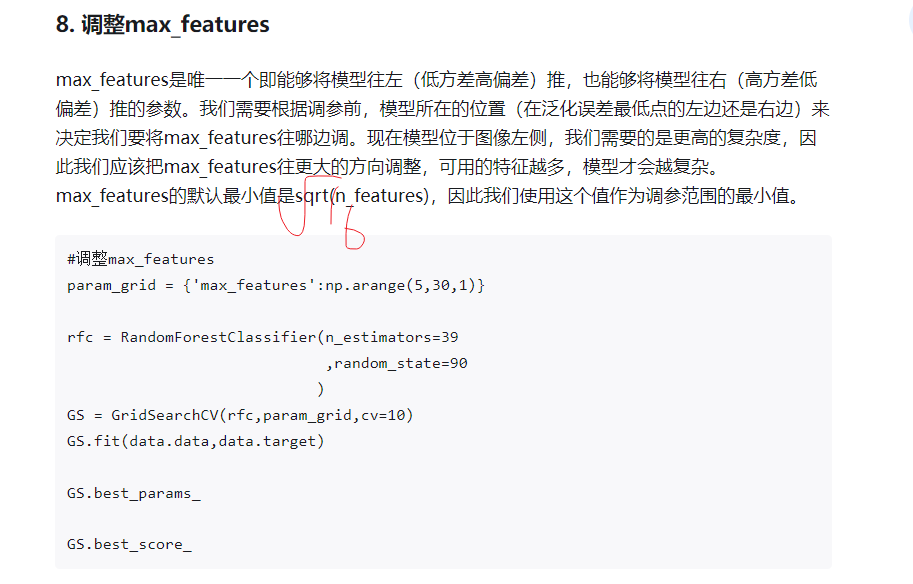
1-50

#{'max\_depth': 21}----0.9843704224283165

param\_grid = {

'max\_depth': np.arange(1,51,1), # 调整树的深度

}



4-30

#{'max\_features': 4}---0.9908702864667346

param\_grid = {

'max\_features': np.arange(4,31,1), # 调整树的深度

}



1-50

{'min\_samples\_leaf': 1}----0.9908702864667346---不变

默认值

param\_grid = {

'min\_samples\_leaf': np.arange(1,51,1), # 调整树的深度

}

在scikit-learn中，**min\_samples\_leaf**的默认值是1，即叶子节点上至少需要包含一个样本。这是为了确保每个叶子节点都包含有关训练数据的信息，避免过拟合。



2-21

#{'min\_samples\_split': 2}---0.9908702864667346---不变

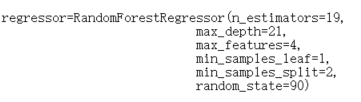
默认值

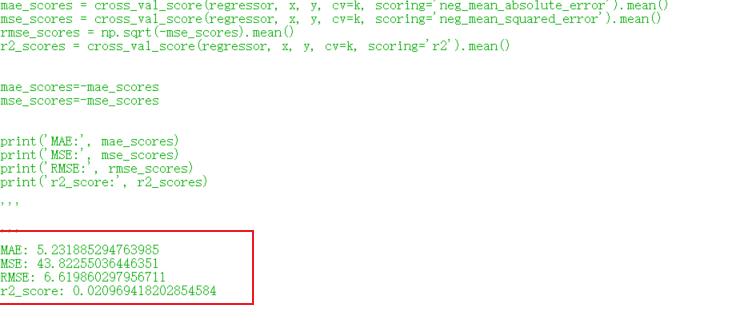
param\_grid = {

'min\_samples\_split': np.arange(2,22,1), # 调整树的深度

}

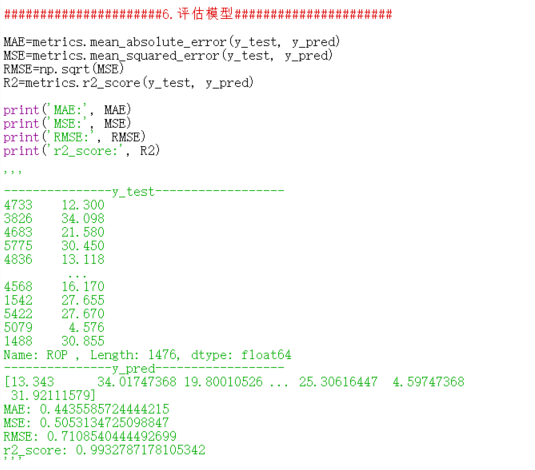
在 scikit-learn 中，**min\_samples\_split** 的默认值是 2，即每个节点至少需要包含两个样本才能继续分裂。这是为了确保至少有两个样本用于构建子节点，避免在样本量较小时过于细致的分裂。



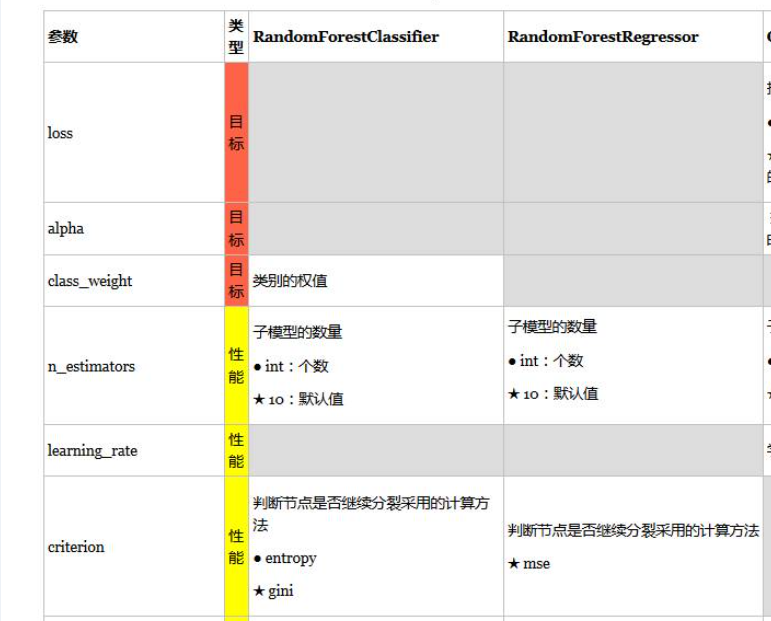


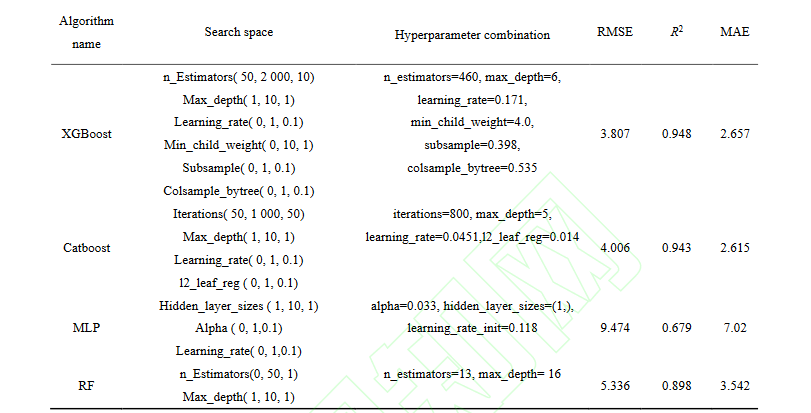
还是垃圾---交叉验证

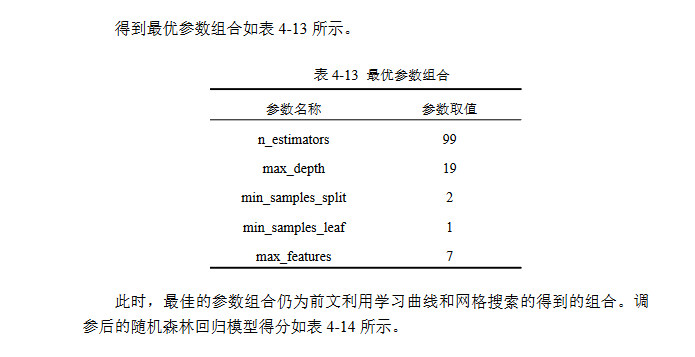
不交叉验证，就验证一次测试集就不垃圾---再调调其他模型吧



<https://cloud.tencent.com/developer/article/1087035>







criterion不了吧，回归问题RF



性能指标的选择通常取决于具体的任务和数据集。在回归预测中，常见的性能指标包括均方误差（Mean Squared Error，MSE）、均方根误差（Root Mean Squared Error，RMSE）、平均绝对误差（Mean Absolute Error，MAE）等。

对于性能指标的评估，一般有两种主要的方式：

1. \*\*在测试集上的评估：\*\* 将数据集划分为训练集和测试集，模型在测试集上进行预测，然后使用选定的性能指标评估模型在测试集上的表现。这是常见的方式，特别是在实际应用中，我们通常关注模型在未见过的新数据上的表现。

2. \*\*使用交叉验证：\*\* 将数据集分成多个折（folds），每次将其中一折作为测试集，其余的作为训练集，多次训练和测试，最后取平均性能指标。这种方法可以更全面地评估模型的性能，尤其在数据量较小或者希望更可靠的性能评估时使用。

选择哪种方式取决于你的任务需求和可用的数据。在实际中，通常会采用第一种方式，即在独立的测试集上评估模型性能。然而，对于数据有限的情况，或者希望更可靠的评估时，使用交叉验证可能更合适。

在 scikit-learn 中，可以使用 `cross\_val\_score` 函数进行交叉验证，也可以使用 `train\_test\_split` 划分数据集来在测试集上评估性能。

