

Hasil dan Analisa :

- Iterasi 100 dan 50 pada RS menghasilkan nilai AUC yang sama yaitu 0.695. Hanya berbeda sedikit dibandingkan menggunakan GS yaitu 0.738. Waktu komputasi berbeda cukup jauh, untuk GS diperlukan 82 menit dengan nilai AUC 0.738 sedangkan dengan RS diperlukan 22 menit dengan nilai AUC 0.695
- Nilai RS tidak mencapai nilai GS karena nilai hyperparameter dengan AUC tertinggi pada GS tidak masuk ke dalam hyperparameter search space RS

Code:

```
In [1]: import pandas as pd
import numpy as np
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from xgboost import XGBClassifier
```

```
In [2]: X_bank=pd.read_csv('dataset/X_bank_preprocessed.csv').to_numpy()
y_bank=pd.read_csv('dataset/y_bank_preprocessed.csv').to_numpy().ravel()
```

Model dan Parameter

```
In [3]: model=XGBClassifier()
```

```
In [38]: params={
    'eta': list(np.linspace(0.001,1,10)), # Learning rate
    'subsample': list(np.linspace(0,1,10)),
    'max_depth': list(range(5,50+1,5)),
    'gamma': list(np.linspace(0,1,10)),
    'min_child_weight': list(range(0,15+1))
}
```

```
In [12]: print('eta : ',params['eta'],'\n') # Learning rate
print('subsample : ',params['subsample'],'\n')
print('max_depth : ',params['max_depth'],'\n')
print('gamma : ',params['gamma'],'\n')
print('min_child_weight : ', params['min_child_weight'],'\n')

eta :  [0.001, 0.112, 0.223, 0.334, 0.445, 0.556, 0.667, 0.778, 0.889, 1.0]

subsample :  [0.0, 0.111111111111111, 0.222222222222222, 0.333333333333333
3, 0.444444444444444, 0.555555555555556, 0.666666666666666, 0.777777777777
777, 0.888888888888888, 1.0]

max_depth :  [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

gamma :  [0.0, 0.111111111111111, 0.222222222222222, 0.333333333333333, 0.
444444444444444, 0.555555555555556, 0.666666666666666, 0.777777777777777,
0.888888888888888, 1.0]

min_child_weight :  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15]
```

```
In [4]: params2={
    'eta': list(np.linspace(0.001,1,100)), # Learning rate
    'subsample': list(np.linspace(0,1,20)),
    'max_depth': list(range(5,50+1,5)),
    'gamma': list(np.linspace(0,1,50)),
    'min_child_weight': list(range(0,15+1))
}
```

```
In [5]: print('eta : ',params2['eta'],'\n') # Learning rate
print('subsample : ',params2['subsample'],'\n')
print('max_depth : ',params2['max_depth'],'\n')
print('gamma : ',params2['gamma'],'\n')
print('min_child_weight : ', params2['min_child_weight'],'\n')
```

eta : [0.001, 0.011090909090909092, 0.021181818181818184, 0.031272727272727272, 0.041363636363636366, 0.051454545454545456, 0.06154545454545455, 0.07163636363636364, 0.08172727272727273, 0.09181818181818183, 0.10190909090909092, 0.112, 0.1220909090909091, 0.13218181818181818, 0.14227272727272727, 0.15236363636363637, 0.16245454545454546, 0.17254545454545456, 0.18263636363636365, 0.19272727272727275, 0.20281818181818184, 0.2129090909090909, 0.223, 0.2330909090909091, 0.2431818181818182, 0.25327272727272726, 0.26336363636363636, 0.27345454545454545, 0.28354545454545454, 0.29363636363636364, 0.30372727272727273, 0.31381818181818183, 0.3239090909090909, 0.334, 0.3440909090909091, 0.3541818181818182, 0.3642727272727273, 0.3743636363636364, 0.3844545454545455, 0.3945454545454546, 0.4046363636363637, 0.4147272727272727, 0.4248181818181818, 0.4349090909090909, 0.445, 0.4550909090909091, 0.4651818181818182, 0.4752727272727273, 0.4853636363636364, 0.4954545454545455, 0.5055454545454545, 0.5156363636363637, 0.5257272727272727, 0.5358181818181819, 0.5459090909090909, 0.556, 0.5660909090909091, 0.5761818181818182, 0.5862727272727273, 0.5963636363636364, 0.6064545454545455, 0.6165454545454546, 0.6266363636363637, 0.6367272727272728, 0.6468181818181818, 0.6569090909090909, 0.667, 0.6770909090909091, 0.6871818181818182, 0.6972727272727273, 0.7073636363636364, 0.7174545454545455, 0.7275454545454546, 0.7376363636363636, 0.7477272727272728, 0.7578181818181818, 0.7679090909090909, 0.778, 0.7880909090909092, 0.7981818181818182, 0.8082727272727274, 0.8183636363636364, 0.8284545454545454, 0.8385454545454546, 0.8486363636363636, 0.8587272727272728, 0.8688181818181818, 0.8789090909090909, 0.889, 0.8990909090909092, 0.9091818181818182, 0.9192727272727274, 0.9293636363636364, 0.9394545454545455, 0.9495454545454546, 0.9596363636363637, 0.9697272727272728, 0.9798181818181818, 0.9899090909090909, 1.0]

subsample : [0.0, 0.05263157894736842, 0.10526315789473684, 0.15789473684210525, 0.21052631578947367, 0.2631578947368421, 0.3157894736842105, 0.3684210526315789, 0.42105263157894735, 0.47368421052631576, 0.5263157894736842, 0.5789473684210527, 0.631578947368421, 0.6842105263157894, 0.7368421052631579, 0.7894736842105263, 0.8421052631578947, 0.894736842105263, 0.9473684210526315, 1.0]

max_depth : [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

gamma : [0.0, 0.02040816326530612, 0.04081632653061224, 0.06122448979591836, 0.08163265306122448, 0.1020408163265306, 0.12244897959183673, 0.14285714285714285, 0.16326530612244897, 0.18367346938775508, 0.2040816326530612, 0.22448979591836732, 0.24489795918367346, 0.26530612244897955, 0.2857142857142857, 0.3061224489795918, 0.32653061224489793, 0.3469387755102041, 0.36734693877551017, 0.3877551020408163, 0.4081632653061224, 0.42857142857142855, 0.44897959183673464, 0.4693877551020408, 0.4897959183673469, 0.5102040816326531, 0.5306122448979591, 0.5510204081632653, 0.5714285714285714, 0.5918367346938775, 0.6122448979591836, 0.6326530612244897, 0.6530612244897959, 0.673469387755102, 0.6938775510204082, 0.7142857142857142, 0.7346938775510203, 0.7551020408163265, 0.7755102040816326, 0.7959183673469387, 0.8163265306122448, 0.836734693877551, 0.8571428571428571, 0.8775510204081632, 0.8979591836734693, 0.9183673469387754, 0.9387755102040816, 0.9591836734693877, 0.9795918367346939, 1.0]

min_child_weight : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

Experiment 1

- Iterasi=200
- random state=1

```
In [6]: iteration=200
auc=[]
std=[]

random_search=RandomizedSearchCV(
    model,
    params2,
    n_iter=iteration,
    scoring='roc_auc',
    n_jobs=-1, cv=5,
    random_state=1)
random_search.fit(X_bank,y_bank)

best_index=random_search.best_index_
auc.append(random_search.cv_results_['mean_test_score'])
std.append(random_search.cv_results_['std_test_score'])
best_param=random_search.best_params_
```

```
In [7]: print("Iterasi: ",iteration)
        print("All AUC: ",auc[0],'\n')
        print(np.mean(auc[0]))
        print("All std: ",std[0],'\n')
        print("Best Hyperparameter: ",best_param,'\n')
        print("Best index/iterasi : ",best_index)
        print("Best AUC :",auc[0][best_index],"( std:",std[0][best_index],")",'\n')
```

Iterasi: 200

All AUC: [0.53348124 0.49785851 0.5 0.5 0.64739294 0.6731435
 0.49143318 0.70773285 0.58892788 0.51328746 0.5 0.51173498
 0.57243581 0.47250092 0.51666158 0.56237815 0.57097184 0.5063618
 0.53074191 0.5 0.50532118 0.53761297 0.57215295 0.5
 0.48897628 0.54696601 0.52357493 0.58251458 0.55438695 0.47635857
 0.61790796 0.52779491 0.5527723 0.59081176 0.50857256 0.58169562
 0.58205232 0.51170686 0.5399496 0.59530544 0.52262516 0.57954928
 0.58779947 0.49488048 0.55229454 0.50901669 0.50278291 0.52250972
 0.5 0.52552846 0.50615812 0.51245741 0.47793607 0.62381461
 0.52073378 0.52401703 0.51027271 0.60593124 0.59663425 0.54456065
 0.61522565 0.51117102 0.50511311 0.56839844 0.54315057 0.5
 0.57096509 0.53673719 0.54382208 0.5 0.63431218 0.56882867
 0.54369241 0.71545875 0.52309331 0.54916963 0.55542578 0.63533685
 0.55005559 0.49869061 0.50062747 0.49663512 0.59371213 0.63967308
 0.51566973 0.50514839 0.72036085 0.51213222 0.51522888 0.60557275
 0.53571929 0.6232049 0.55726112 0.49253379 0.50439629 0.58463796
 0.63846797 0.48920397 0.50695279 0.59190371 0.49653566 0.68557838
 0.51221953 0.56270605 0.54048358 0.49944633 0.53609028 0.51993061
 0.58448148 0.55340333 0.52878731 0.50546439 0.5096 0.48806865
 0.5434078 0.50081519 0.52453595 0.47394302 0.47895572 0.54420524
 0.51776349 0.5 0.4815376 0.51691493 0.59787901 0.53225126
 0.64410554 0.52945067 0.5 0.48332944 0.53604407 0.68218789
 0.49413363 0.5550185 0.49180991 0.50343397 0.5 0.53390981
 0.51833181 0.49081071 0.5 0.55825061 0.63772859 0.5
 0.5451513 0.56887762 0.60688308 0.52760726 0.56224662 0.50736216
 0.51884321 0.50503066 0.55634494 0.57887115 0.52717886 0.47783647
 0.55083622 0.51714446 0.5 0.55125785 0.57524731 0.48340726
 0.51736399 0.48509502 0.52976497 0.51567547 0.49704708 0.70758506
 0.52407001 0.54544373 0.533061 0.62194196 0.48548371 0.47839226
 0.5 0.59621766 0.57359608 0.51177658 0.5131502 0.50244435
 0.63843126 0.47357082 0.5781774 0.67070166 0.54945649 0.49568402
 0.61613651 0.53639083 0.64705734 0.5794642 0.50354075 0.55428652
 0.53823129 0.69685588 0.58453057 0.57526952 0.56635083 0.51243977
 0.4874274 0.64498743]

0.5447572310215225

All std: [0.10929969 0.09808629 0. 0. 0.11676919 0.08868502
 0.08529388 0.07450728 0.11988682 0.09822186 0. 0.10295943
 0.10679342 0.0991513 0.09319521 0.09401297 0.10650611 0.09977693
 0.11303552 0. 0.09432527 0.07952898 0.14045341 0.
 0.08317535 0.10095832 0.10747176 0.10496503 0.08238913 0.09039859
 0.1260276 0.10672685 0.04718328 0.06703747 0.08234331 0.09030407
 0.11490687 0.11009411 0.09396971 0.12058881 0.07952849 0.09180459
 0.1187925 0.0967493 0.09784742 0.09148605 0.09044518 0.11250528
 0. 0.09154573 0.08324493 0.08569666 0.1147312 0.10687003
 0.08883511 0.08396378 0.08979044 0.10329093 0.10261953 0.07620044
 0.12032961 0.08173941 0.10832186 0.13322301 0.10334082 0.
 0.08676841 0.10399185 0.09321615 0. 0.14086561 0.10328538
 0.09871244 0.07651684 0.10879194 0.05236358 0.12301839 0.11220622
 0.06692172 0.09261423 0.0982469 0.12348244 0.11182354 0.07979162
 0.10721631 0.09158396 0.07185907 0.09632836 0.08088696 0.13044159
 0.11580772 0.12431985 0.08714445 0.09750125 0.09576199 0.12519105
 0.11936783 0.08902269 0.10623799 0.12624642 0.08071832 0.12889513
 0.08358865 0.12453248 0.1300323 0.11222581 0.09931707 0.07745766
 0.10798511 0.07569076 0.09175096 0.10115844 0.08049888 0.08005746
 0.082461 0.1024338 0.07178428 0.08177185 0.09304377 0.05679759]

```

0.08846015 0. 0.08392977 0.08322557 0.10339147 0.09774966
0.11510238 0.09298489 0. 0.08454762 0.07260783 0.13570425
0.08515228 0.08974789 0.08206251 0.09585841 0. 0.10263373
0.08635342 0.10376268 0. 0.12637524 0.12480721 0.
0.10612181 0.056662 0.13759659 0.09046051 0.11115563 0.10225686
0.09611559 0.08072129 0.12488491 0.13605162 0.10483281 0.09704402
0.11198263 0.10010623 0. 0.06363705 0.09621165 0.08949732
0.10086536 0.10833375 0.11148525 0.08701586 0.10182292 0.11944756
0.09534248 0.09911793 0.08679485 0.08805132 0.094797 0.09035092
0. 0.0861153 0.1145713 0.09875569 0.0706163 0.08542738
0.10786514 0.09072099 0.1042639 0.13188845 0.09836704 0.10658383
0.11933281 0.10962013 0.14494977 0.12246822 0.08895589 0.10460761
0.09729067 0.0990915 0.12114987 0.11320942 0.12002646 0.10440358
0.08052384 0.0964038 ]

```

Best Hyperparameter: {'subsample': 0.05263157894736842, 'min_child_weight': 14, 'max_depth': 45, 'gamma': 0.9183673469387754, 'eta': 0.8183636363636364}

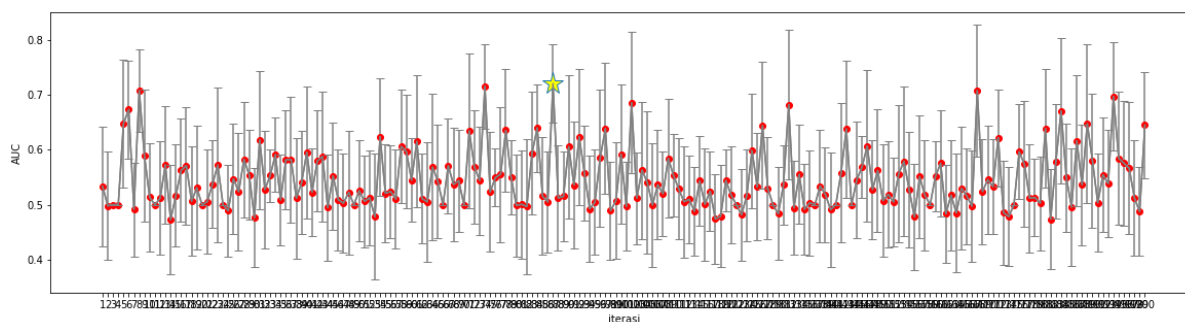
Best index/iterasi : 86

Best AUC : 0.7203608516313473 (std: 0.07185906796685554)

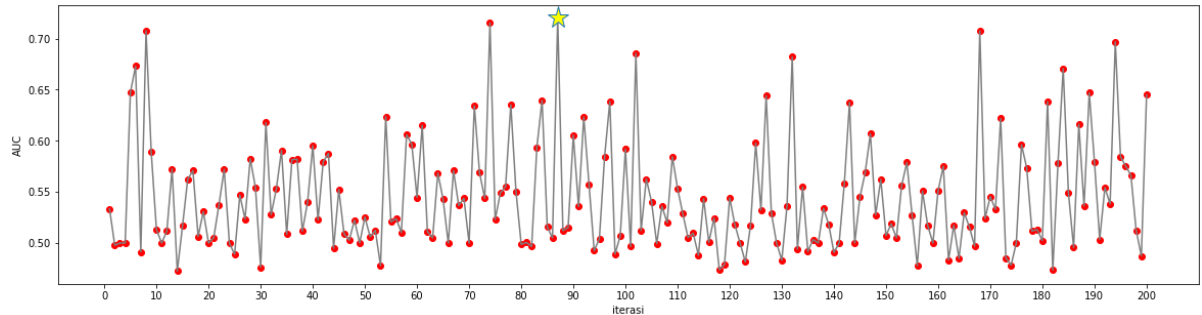
```

In [23]: import matplotlib.pyplot as plt
fig=plt.figure(figsize=(20,5))
x=range(1,iteration+1)
y=auc[0]
error=std[0]
plt.plot(x,y,'or')
plt.plot(x,y,color='gray')
plt.plot(x[best_index],y[best_index],marker='*',markersize=22,markerfacecolor='yellow')
plt.errorbar(x, y, yerr=error,color='gray',capsize=4)
fig=plt.xticks(range(1,iteration+1))
fig=plt.xlabel('iterasi')
fig=plt.ylabel('AUC')

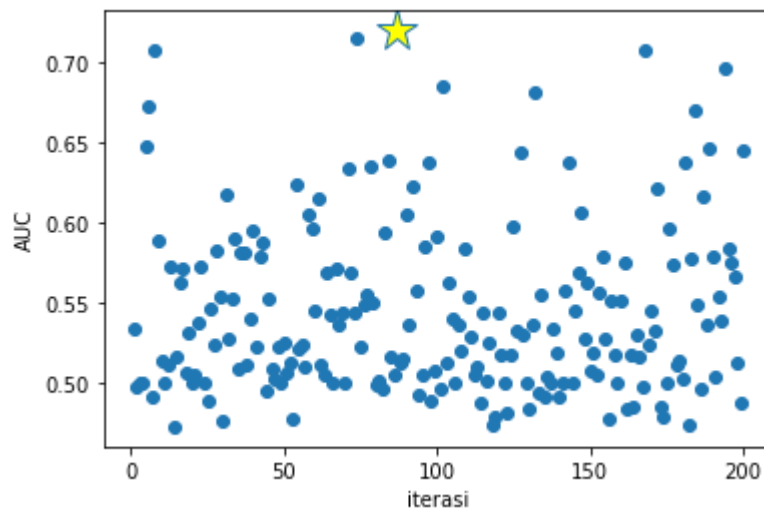
```



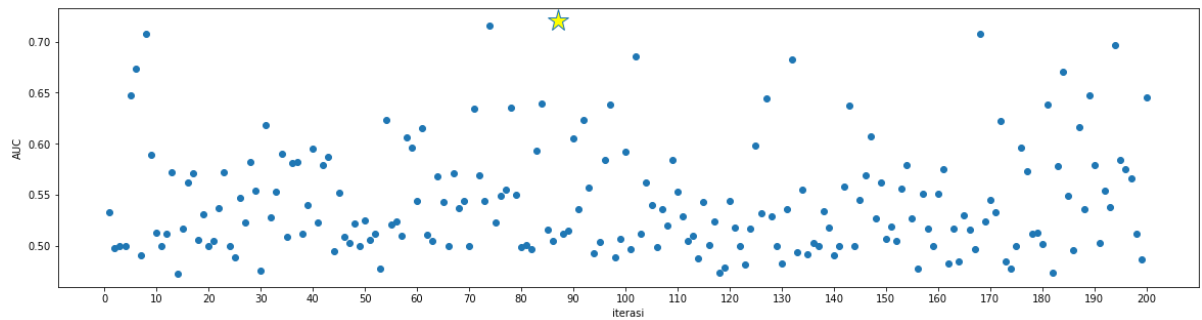

```
In [21]: fig=plt.figure(figsize=(20,5))
x=range(1,iteration+1)
y=auc[0]
error=std[0]
plt.plot(x,y,'or')
plt.plot(x,y,color='gray')
plt.plot(x[best_index],y[best_index],marker='*',markersize=22,markerfacecolor='yellow')
fig=plt.xticks(range(1,iteration+1))
fig=plt.xlabel('iterasi')
fig=plt.ylabel('AUC')
fig=plt.xticks(range(0,210,10))
```



```
In [17]: fig=plt.figure()
x=range(1,iteration+1)
y=auc[0]
error=std[0]
plt.scatter(x,y)
plt.plot(x[best_index],y[best_index],marker='*',markersize=22,markerfacecolor='yellow')
fig=plt.xticks(range(1,iteration+1))
fig=plt.xlabel('iterasi')
fig=plt.ylabel('AUC')
fig=plt.xticks(range(0,210,50))
```



```
In [22]: fig=plt.figure(figsize=(20,5))
x=range(1,iteration+1)
y=auc[0]
error=std[0]
plt.scatter(x,y)
plt.plot(x[best_index],y[best_index],marker='*',markersize=22,markerfacecolor='yellow')
fig=plt.xticks(range(1,iteration+1))
fig=plt.xlabel('iterasi')
fig=plt.ylabel('AUC')
fig=plt.xticks(range(0,210,10))
```



Experiment 2

- random search with early stopping

```
In [ ]: iteration=[10,20,30,40,50,60,70,80,90,100,110,120,130,140,150]
for i in iteration:
    rnd_srch_clf=RandomizedSearchCV(
        model,
        params,
        n_iter=i,
        scoring='roc_auc',
        n_jobs=-1, cv=5,
        random_state=1)
```

```
In [ ]: rnd_srch_clf=RandomizedSearchCV(
        model,
        params,
        n_iter=50,
        scoring='roc_auc',
        n_jobs=-1, cv=5,
        random_state=1)
```

```
In [ ]: index=rnd_srch_clf.best_index_
print()
print("Best params: ",rnd_srch_clf.best_params_)
print("AUC terbaik: ",rnd_srch_clf.cv_results_['mean_test_score'][index])
print("std terbaik: ",rnd_srch_clf.cv_results_['std_test_score'][index])
print("Semua AUC: ",rnd_srch_clf.cv_results_['mean_test_score'])
print("Semua std: ",rnd_srch_clf.cv_results_['std_test_score'][index])
```

```
In [30]: rnd_srch_clf=RandomizedSearchCV(
            model,
            params,
            n_iter=50,
            scoring='roc_auc',
            n_jobs=-1, cv=5,
            random_state=1)
```

```
In [31]: rnd_srch_clf.fit(X_bank,y_bank)
```

```
Out[31]: RandomizedSearchCV(cv=5, error_score=nan,
                             estimator=XGBClassifier(base_score=None, booster=None,
                                                       colsample_bylevel=None,
                                                       colsample_bynode=None,
                                                       colsample_bytree=None, gamma=None,
                                                       gpu_id=None, importance_type='gain',
                                                       interaction_constraints=None,
                                                       learning_rate=None,
                                                       max_delta_step=None, max_depth=None,
                                                       min_child_weight=None, missing=nan,
                                                       monotone_constraints=None,
                                                       n_estimators=50,
                                                       num_parallel_tree=None,
                                                       random_state=None,
                                                       reg_lambda=None,
                                                       subsample=None,
                                                       verbose=0),
                             n_jobs=-1,
                             n_iter=50,
                             pre_dispatch='2*n_jobs',
                             random_state=1,
                             refit=True,
                             return_train_score=False,
                             scoring='roc_auc',
                             verbose=0)
```

```
In [32]: index=rnd_srch_clf.best_index_  
print()  
print("Best params: ",rnd_srch_clf.best_params_)  
print("AUC terbaik: ",rnd_srch_clf.cv_results_['mean_test_score'][index])  
print("std terbaik: ",rnd_srch_clf.cv_results_['std_test_score'][index])  
print("Semua AUC: ",rnd_srch_clf.cv_results_['mean_test_score'])  
print("Semua std: ",rnd_srch_clf.cv_results_['std_test_score'][index])
```

Best params:

```
{'subsample': 0.2222222222222222, 'min_child_weight': 10, 'max_depth': 20, 'gamma': 0.6666666666666666, 'eta': 0.001}
```

AUC:

```
0.6957506053674708
```

std:

```
0.15371171277770676
```

```
In [35]: rnd_srch_clf_2=RandomizedSearchCV(  
        model,  
        params,  
        n_iter=100,  
        scoring='roc_auc',  
        n_jobs=-1, cv=5,  
        random_state=1)
```

```
In [36]: rnd_srch_clf_2.fit(X_bank,y_bank)
```

```
Out[36]: RandomizedSearchCV(cv=5, error_score=nan,
                             estimator=XGBClassifier(base_score=None, booster=None,
                                                       colsample_bylevel=None,
                                                       colsample_bynode=None,
                                                       colsample_bytree=None, gamma=None,
                                                       gpu_id=None, importance_type='gai
n',
                                                       interaction_constraints=None,
                                                       learning_rate=None,
                                                       max_delta_step=None, max_depth=Non
e,
                                                       min_child_weight=None, missing=na
n,
                                                       monotone_constraints=None,
                                                       n...
                             'max_depth': [5, 10, 15, 20, 25, 30,
                                             35,
                                             40, 45, 50],
                             'min_child_weight': [0, 1, 2, 3, 4,
                                                    5,
                                                    6, 7, 8, 9, 10,
                                                    11,
                                                    12, 13, 15],
                             'subsample': [0.0, 0.11111111111111111
                                             0.22222222222222222,
                                             0.33333333333333333,
                                             0.44444444444444444,
                                             0.55555555555555556,
                                             0.66666666666666666,
                                             0.77777777777777777,
                                             0.88888888888888888,
                                             1.0]},
                             pre_dispatch='2*n_jobs', random_state=1, refit=True,
                             return_train_score=False, scoring='roc_auc', verbose=0)
```

```
In [37]: index=rnd_srch_clf_2.best_index_
print("Best params: ")
print(rnd_srch_clf_2.best_params_)
print("AUC: ")
print(rnd_srch_clf_2.cv_results_['mean_test_score'][index])
print("std: ")
print(rnd_srch_clf_2.cv_results_['std_test_score'][index])
```

```
Best params:
{'subsample': 0.22222222222222222, 'min_child_weight': 10, 'max_depth': 20, 'g
amma': 0.66666666666666666, 'eta': 0.001}
AUC:
0.6957506053674708
std:
0.15371171277770676
```