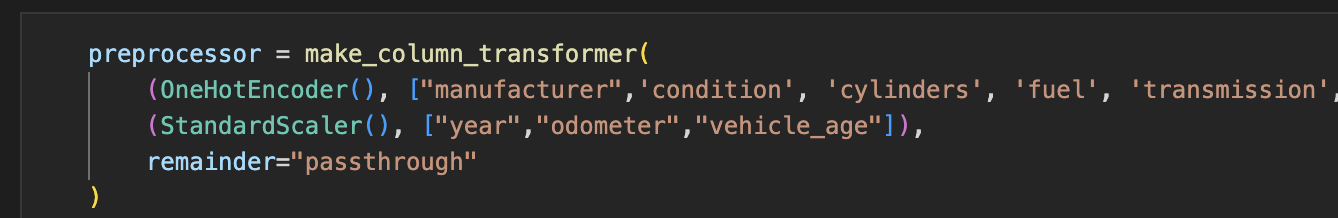
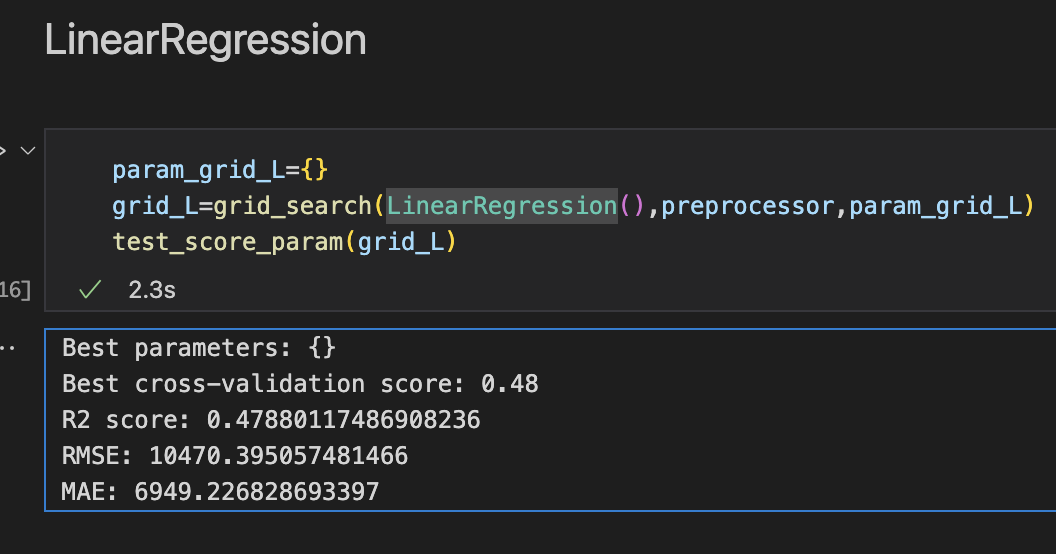
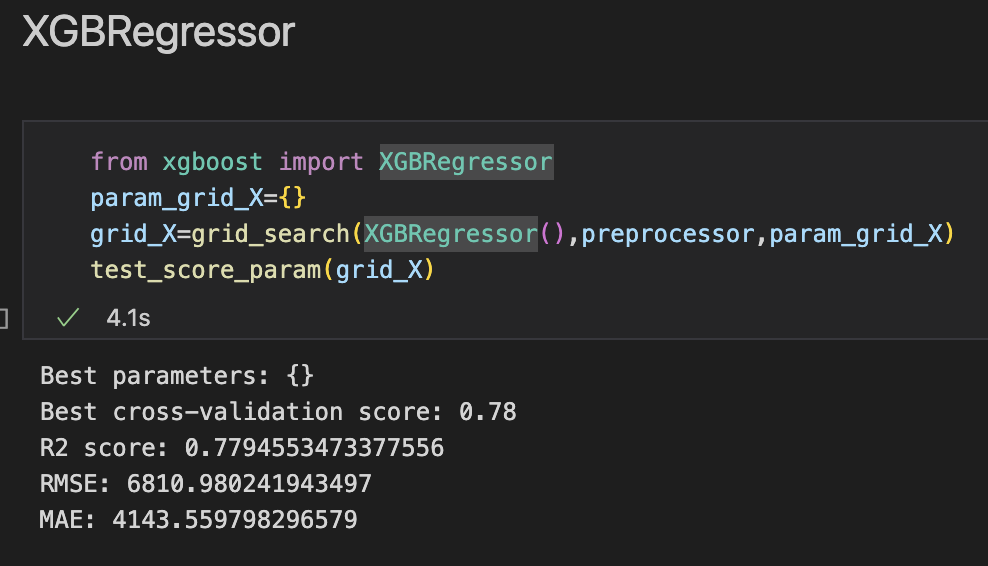
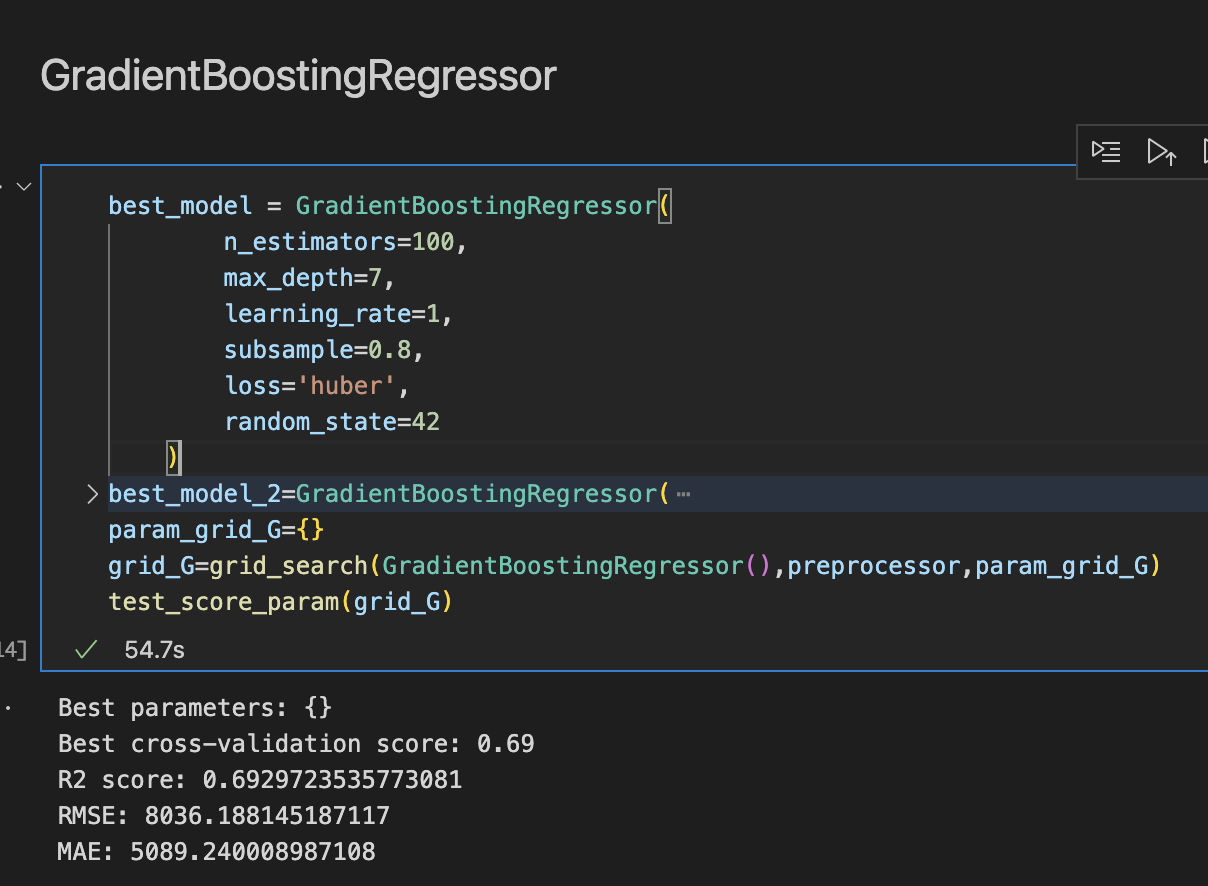
## 初始工作

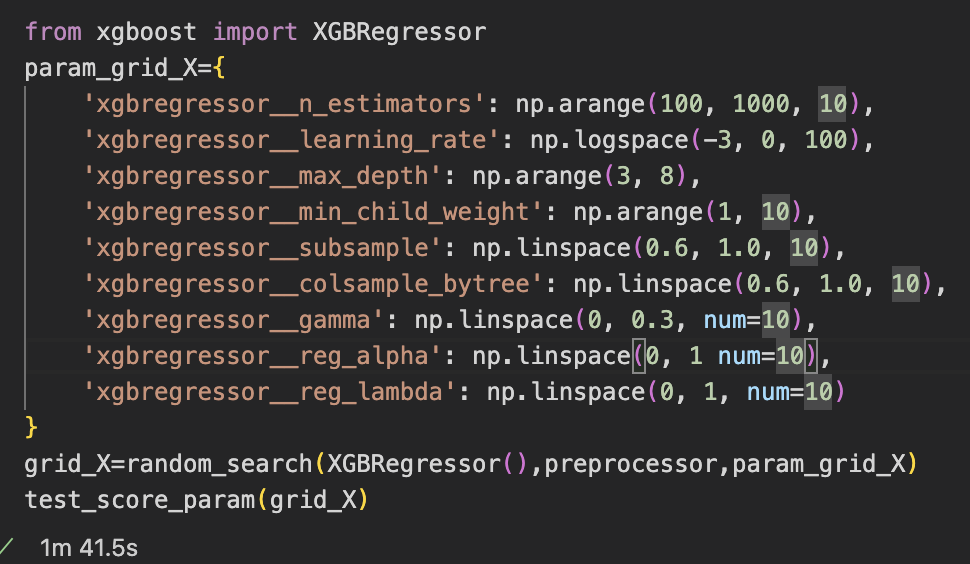
之前在data preprocess时对参数进行了初步的encoding

之后对数据急性二次处理，包括normalization和one hot 编码：

在未使用randomsearch和gridsearch对sklearn的模型做最佳超参数的搜寻的情况下，各个回归模型在数据的表现：



## XGBRegressor

从第一次实验可知：XGBRegressor速度快且，表现较好，所以我们先针对该模型进行最佳超参数搜寻，先采用RandomSearch快速搜寻最佳的超参数：

Iter（迭代次数）=10 的情况下的输出结果为：

Best parameters: {'xgbregressor\_\_subsample': 0.8666666666666667, 'xgbregressor\_\_reg\_lambda': 0.5555555555555556, 'xgbregressor\_\_reg\_alpha': 1.0,

'xgbregressor\_\_n\_estimators': 780, 'xgbregressor\_\_min\_child\_weight': 4,

'xgbregressor\_\_max\_depth': 6,

'xgbregressor\_\_learning\_rate': 0.037649358067924674, 'xgbregressor\_\_gamma': 0.06666666666666667, 'xgbregressor\_\_colsample\_bytree': 0.8222222222222222}

Best cross-validation score: 0.78

R2 score: 0.7814601094624093

RMSE: 6779.953493663403

MAE: 4115.321202050043

模型性能有细微的提升，再对超参数进行一些微调：

重新运行iter为10的randomsearch：

Best parameters: {

'xgbregressor\_\_subsample': 0.9555555555555555, 'xgbregressor\_\_reg\_lambda': 0.8888888888888888, 'xgbregressor\_\_reg\_alpha': 50,

'xgbregressor\_\_n\_estimators': 820, 'xgbregressor\_\_min\_child\_weight': 6,

'xgbregressor\_\_max\_depth': 7,

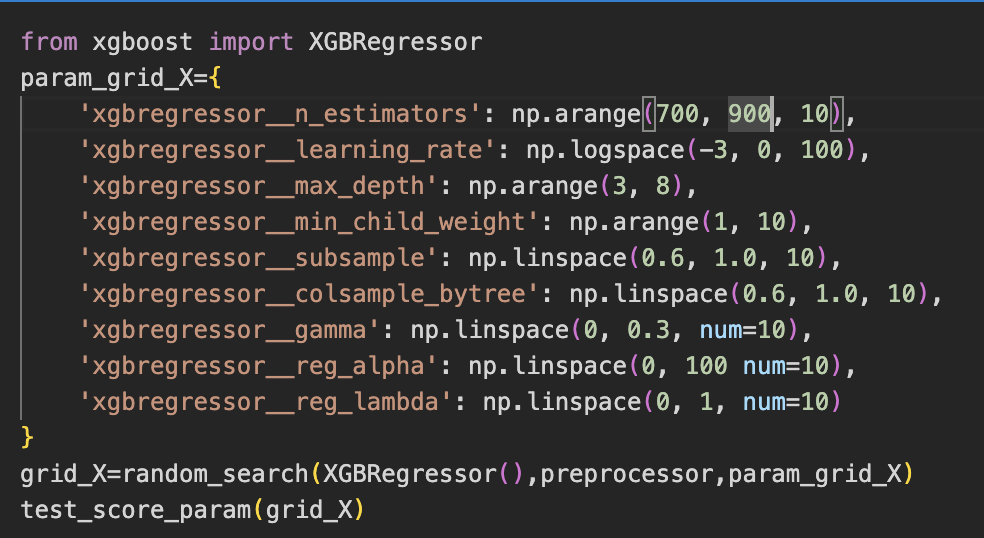
'xgbregressor\_\_learning\_rate': 0.49770235643321137, 'xgbregressor\_\_gamma': 0.26666666666666666, 'xgbregressor\_\_colsample\_bytree': 0.9555555555555555}

Best cross-validation score: 0.83

R2 score: 0.8337832108996921

RMSE: 5912.876369346471

将iter调整为20，运行结果为：

Best parameters: {

'xgbregressor\_\_subsample': 0.7333333333333333, 'xgbregressor\_\_reg\_lambda': 0.8888888888888888, 'xgbregressor\_\_reg\_alpha': 40,

'xgbregressor\_\_n\_estimators': 840, 'xgbregressor\_\_min\_child\_weight': 2,

'xgbregressor\_\_max\_depth': 7,

'xgbregressor\_\_learning\_rate': 0.32745491628777285, 'xgbregressor\_\_gamma': 0.26666666666666666, 'xgbregressor\_\_colsample\_bytree': 0.7333333333333333}

Best cross-validation score: 0.83

R2 score: 0.8321239502607707

RMSE: 5942.315753658962

MAE: 3342.263672206873

模型性能提升较为明显，

把iter调整为30，运行结果为：

Best parameters: {

'xgbregressor\_\_subsample': 0.9111111111111111, 'xgbregressor\_\_reg\_lambda': 0.3333333333333333, 'xgbregressor\_\_reg\_alpha': 80,

'xgbregressor\_\_n\_estimators': 760, 'xgbregressor\_\_min\_child\_weight': 9,

'xgbregressor\_\_max\_depth': 6,

'xgbregressor\_\_learning\_rate': 0.24770763559917114, 'xgbregressor\_\_gamma': 0.3,

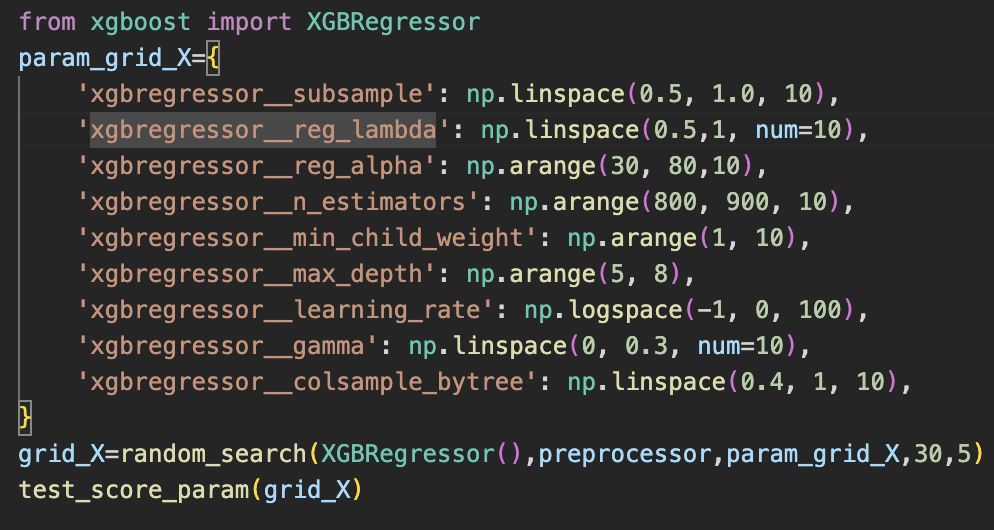
'xgbregressor\_\_colsample\_bytree': 0.6444444444444444}

Best cross-validation score: 0.81

R2 score: 0.8160444313509987

RMSE: 6220.3929986253825

MAE: 3657.645487969625

根据三次结果我们可以逐渐缩小超参数的取值范围：

把iter=30，运行结果为：

Best parameters: {

'xgbregressor\_\_subsample': 1.0,

'xgbregressor\_\_reg\_lambda': 0.8333333333333333, 'xgbregressor\_\_reg\_alpha': 70,

'xgbregressor\_\_n\_estimators': 850, 'xgbregressor\_\_min\_child\_weight': 2,

'xgbregressor\_\_max\_depth': 7,

'xgbregressor\_\_learning\_rate': 0.33516026509388425, 'xgbregressor\_\_gamma': 0.0,

'xgbregressor\_\_colsample\_bytree': 0.9333333333333333}

Best cross-validation score: 0.83

R2 score: 0.8401879433892784

RMSE: 5797.838642532647

MAE: 3210.407991550628

iter=50,运行结果为：

Best parameters: {

'xgbregressor\_\_subsample': 0.9444444444444444, 'xgbregressor\_\_reg\_lambda': 0.6111111111111112, 'xgbregressor\_\_reg\_alpha': 50,

'xgbregressor\_\_n\_estimators': 800, 'xgbregressor\_\_min\_child\_weight': 2,

'xgbregressor\_\_max\_depth': 7,

'xgbregressor\_\_learning\_rate': 0.17475284000076838, 'xgbregressor\_\_gamma': 0.13333333333333333, 'xgbregressor\_\_colsample\_bytree': 1.0}

Best cross-validation score: 0.83

R2 score: 0.8335133098183598

RMSE: 6005.368925814105

MAE: 3389.2391193936073

第二次iter=50，运行结果为：

Best parameters: {

'xgbregressor\_\_subsample': 0.8888888888888888, 'xgbregressor\_\_reg\_lambda': 0.8333333333333333, 'xgbregressor\_\_reg\_alpha': 50,

'xgbregressor\_\_n\_estimators': 860, 'xgbregressor\_\_min\_child\_weight': 2,

'xgbregressor\_\_max\_depth': 7,

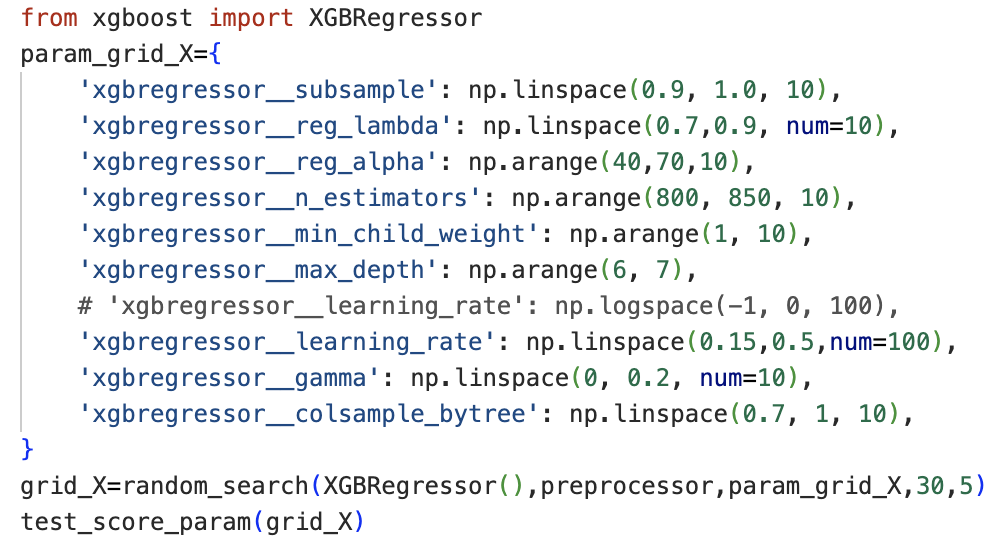
'xgbregressor\_\_learning\_rate': 0.29150530628251775, 'xgbregressor\_\_gamma': 0.13333333333333333, 'xgbregressor\_\_colsample\_bytree': 0.5333333333333333}

Best cross-validation score: 0.83

R2 score: 0.8328508763156843

RMSE: 6017.304439249584

MAE: 3358.4134241359225

由此我们可以再次修改超参数取值范围：

iter=30:

Best parameters: {

'xgbregressor\_\_subsample': 0.9888888888888889, 'xgbregressor\_\_reg\_lambda': 0.8333333333333334, 'xgbregressor\_\_reg\_alpha': 50,

'xgbregressor\_\_n\_estimators': 820, 'xgbregressor\_\_min\_child\_weight': 3,

'xgbregressor\_\_max\_depth': 6,

'xgbregressor\_\_learning\_rate': 0.37626262626262624, 'xgbregressor\_\_gamma': 0.2,

'xgbregressor\_\_colsample\_bytree': 0.9}

Best cross-validation score: 0.83

R2 score: 0.8275738970509248

RMSE: 6111.5509832809985

MAE: 3455.766225851777

iter=30,第二次实验：

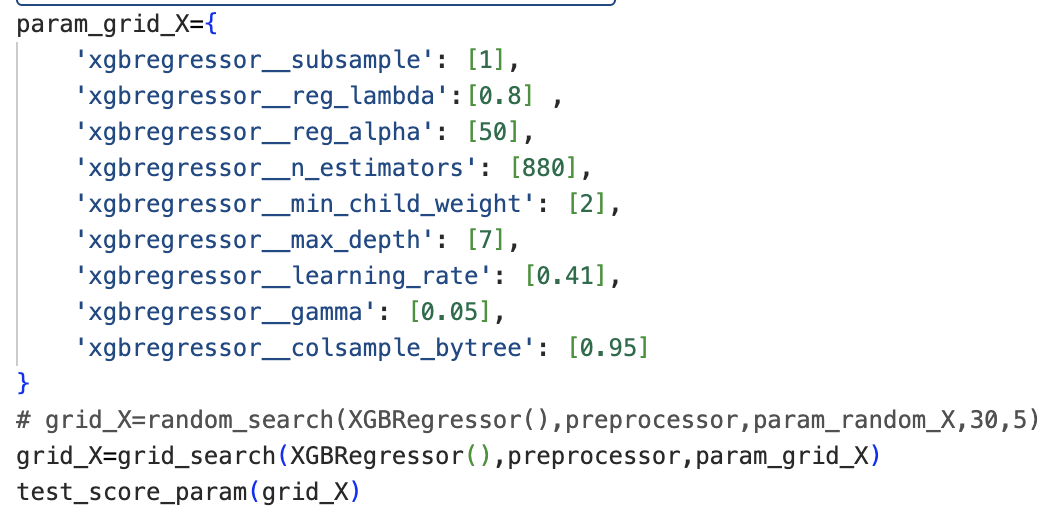
Best parameters: {'xgbregressor\_\_subsample': 0.9777777777777777, 'xgbregressor\_\_reg\_lambda': 0.8, 'xgbregressor\_\_reg\_alpha': 40, 'xgbregressor\_\_n\_estimators': 890, 'xgbregressor\_\_min\_child\_weight': 5, 'xgbregressor\_\_max\_depth': 7, 'xgbregressor\_\_learning\_rate': 0.4131313131313131, 'xgbregressor\_\_gamma': 0.07777777777777778, 'xgbregressor\_\_colsample\_bytree': 0.9666666666666667}

Best cross-validation score: 0.84

R2 score: 0.8371023552746877

RMSE: 5940.285730188645

MAE: 3226.9672052818632

经过多次试验之后我们可以得到一个大概的最佳超参数取值范围，并用GridSearch精准超参数的取值，最后可以得到一个最佳的超参数取值列表：

模型性能测试的结果为：

Best parameters: {

'xgbregressor\_\_colsample\_bytree': 0.95,

'xgbregressor\_\_gamma': 0.05,

'xgbregressor\_\_learning\_rate': 0.41,

'xgbregressor\_\_max\_depth': 7,

'xgbregressor\_\_min\_child\_weight': 2,

'xgbregressor\_\_n\_estimators': 880,

'xgbregressor\_\_reg\_alpha': 50,

'xgbregressor\_\_reg\_lambda': 0.8,

'xgbregressor\_\_subsample': 1}

Best cross-validation score: 0.84

R2 score: 0.8398418059690647

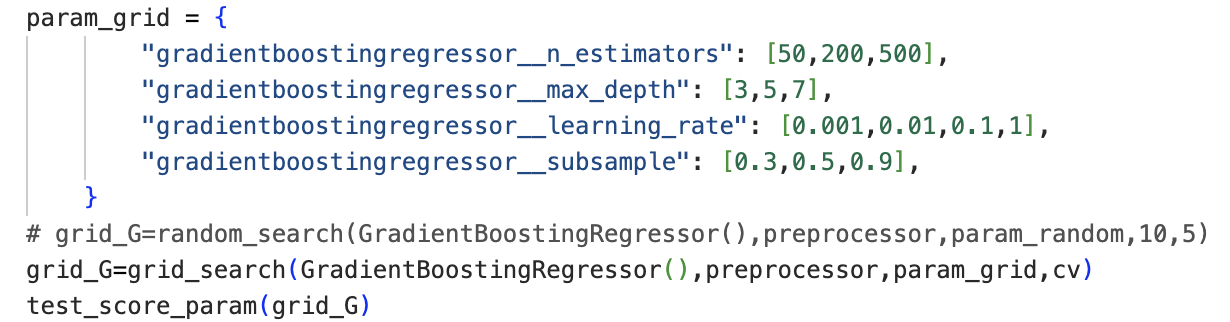
RMSE: 5890.1250370049465

MAE: 3181.194474591849

## GradientBoostingRegressor

接下来对GradientBoostingRegressor寻找最佳超参数：

因为该模型速度较慢，所以我们采用少量设值的方式寻找：

使用GridSearch：

搜寻的结果如下所示，相比于使用默认超参数性能提升较大：

Best parameters: {

'gradientboostingregressor\_\_learning\_rate': 0.1, 'gradientboostingregressor\_\_max\_depth': 7, 'gradientboostingregressor\_\_n\_estimators': 500, 'gradientboostingregressor\_\_subsample': 0.9}

Best cross-validation score: 0.82

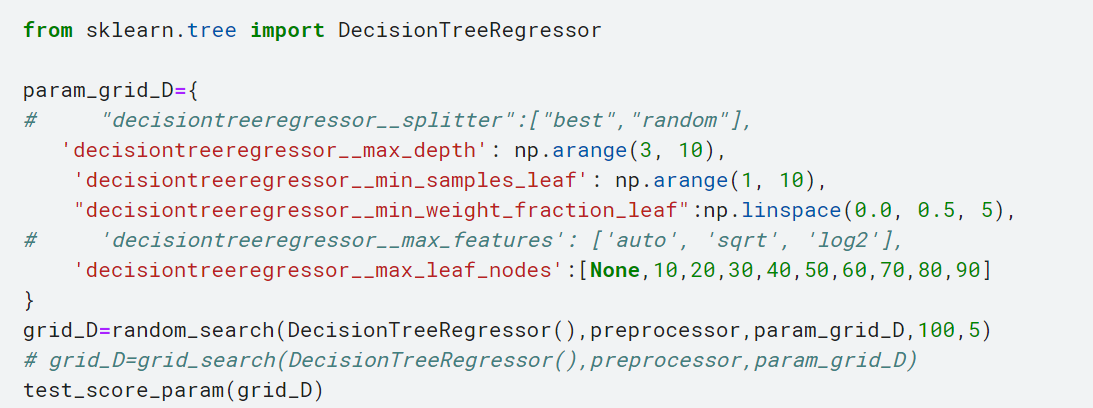
R2 score: 0.8214734502718747

RMSE: 6159.999597470852

MAE: 3686.1962746059808

## Decision Tree Regressor

对于DecisionTreeRegressor：我们可以先用RandomSearch来确认最佳超参数的范围：



我们设置n\_iter=100，输出结果是：

Best parameters: {

'decisiontreeregressor\_\_min\_weight\_fraction\_leaf': 0.0,

'decisiontreeregressor\_\_min\_samples\_leaf': 6,

'decisiontreeregressor\_\_max\_leaf\_nodes': None,

'decisiontreeregressor\_\_max\_depth': 9}

Best cross-validation score: 0.68

R2 score: 0.6807607702966123

RMSE: 8282.790021836507

MAE: 5258.52959258295

让我们多做几次实验：

第2次：

Best parameters: {

'decisiontreeregressor\_\_min\_weight\_fraction\_leaf': 0.0,

'decisiontreeregressor\_\_min\_samples\_leaf': 2,

'decisiontreeregressor\_\_max\_leaf\_nodes': 80,

'decisiontreeregressor\_\_max\_depth': 9}

Best cross-validation score: 0.65

R2 score: 0.6539216826046554

RMSE: 8623.93987136857

MAE: 5616.047386585876

第3次：

Best parameters: {

'decisiontreeregressor\_\_min\_weight\_fraction\_leaf': 0.0,

'decisiontreeregressor\_\_min\_samples\_leaf': 1,

'decisiontreeregressor\_\_max\_leaf\_nodes': None,

'decisiontreeregressor\_\_max\_depth': 8}

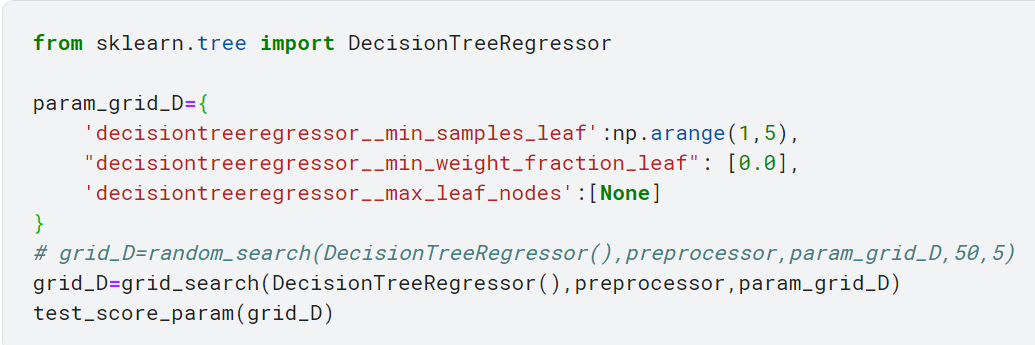
Best cross-validation score: 0.67

R2 score: 0.6609866873016943

RMSE: 8535.45940068161

MAE: 5472.013908617442

可以发现randomsearch选择的最佳超参数比较固定，那么我们直接使用GridSearch进行网格搜索：



Best parameters: {

'decisiontreeregressor\_\_max\_leaf\_nodes': None,

'decisiontreeregressor\_\_min\_samples\_leaf': 4,

'decisiontreeregressor\_\_min\_weight\_fraction\_leaf': 0.0}

Best cross-validation score: 0.79

R2 score: 0.8026535899565279

RMSE: 6512.2811200921005

MAE: 3205.0574951103727