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：

```
preprocessor = make_column_transformer(
    (OneHotEncoder(), ["manufacturer",'condition', 'cylinders', 'fuel', 'transmission',
    (StandardScaler(), ["year","odometer","vehicle_age"]),
    remainder="passthrough"
)
```

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

```
from xgboost import XGBRegressor
param_grid_X={}
grid_X=grid_search(XGBRegressor(),preprocessor,param_grid_X)
test_score_param(grid_X)
```

✓ 4.1s

```
Best parameters: {}
Best cross-validation score: 0.78
R2 score: 0.7794553473377556
RMSE: 6810.980241943497
MAE: 4143.559798296579
```

LinearRegression

```
param_grid_L={}
grid_L=grid_search(LinearRegression(),preprocessor,param_grid_L)
test_score_param(grid_L)
```

✓ 2.3s

```
Best parameters: {}
Best cross-validation score: 0.48
R2 score: 0.47880117486908236
RMSE: 10470.395057481466
MAE: 6949.226828693397
```

DecisionTree Regressor

```python
from sklearn.tree import DecisionTreeRegressor

param_grid_D={}
grid_D=grid_search(DecisionTreeRegressor(),preprocessor,param_grid_D)
test_score_param(grid_D)
```

8] ✓ 1m 29.1s

```
Best parameters: {}
Best cross-validation score: 0.76
R2 score: 0.7796164362180659
RMSE: 6808.492370239102
MAE: 2718.284856456684
```

GradientBoostingRegressor

```python
best_model = GradientBoostingRegressor(
        n_estimators=100,
        max_depth=7,
        learning_rate=1,
        subsample=0.8,
        loss='huber',
        random_state=42
    )
best_model_2=GradientBoostingRegressor( ...
param_grid_G={}
grid_G=grid_search(GradientBoostingRegressor(),preprocessor,param_grid_G)
test_score_param(grid_G)
```

4] ✓ 54.7s

```
Best parameters: {}
Best cross-validation score: 0.69
R2 score: 0.6929723535773081
RMSE: 8036.188145187117
MAE: 5089.240008987108
```

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

```
from xgboost import XGBRegressor
param_grid_X={
    'xgbregressor__n_estimators': np.arange(100, 1000, 10),
    'xgbregressor__learning_rate': np.logspace(-3, 0, 100),
    'xgbregressor__max_depth': np.arange(3, 8),
    'xgbregressor__min_child_weight': np.arange(1, 10),
    'xgbregressor__subsample': np.linspace(0.6, 1.0, 10),
    'xgbregressor__colsample_bytree': np.linspace(0.6, 1.0, 10),
    'xgbregressor__gamma': np.linspace(0, 0.3, num=10),
    'xgbregressor__reg_alpha': np.linspace(0, 1 num=10),
    'xgbregressor__reg_lambda': np.linspace(0, 1, num=10)
}
grid_X=random_search(XGBRegressor(),preprocessor,param_grid_X)
test_score_param(grid_X)
```

✓ 1m 41.5s

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：

```python
from xgboost import XGBRegressor
param_grid_X={
    'xgbregressor__n_estimators': np.arange(700, 900, 10),
    'xgbregressor__learning_rate': np.logspace(-3, 0, 100),
    'xgbregressor__max_depth': np.arange(3, 8),
    'xgbregressor__min_child_weight': np.arange(1, 10),
    'xgbregressor__subsample': np.linspace(0.6, 1.0, 10),
    'xgbregressor__colsample_bytree': np.linspace(0.6, 1.0, 10),
    'xgbregressor__gamma': np.linspace(0, 0.3, num=10),
    'xgbregressor__reg_alpha': np.linspace(0, 100 num=10),
    'xgbregressor__reg_lambda': np.linspace(0, 1, num=10)
}
grid_X=random_search(XGBRegressor(),preprocessor,param_grid_X)
test_score_param(grid_X)
```

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

```
from xgboost import XGBRegressor
param_grid_X={
    'xgbregressor__subsample': np.linspace(0.5, 1.0, 10),
    'xgbregressor__reg_lambda': np.linspace(0.5,1, num=10),
    'xgbregressor__reg_alpha': np.arange(30, 80,10),
    'xgbregressor__n_estimators': np.arange(800, 900, 10),
    'xgbregressor__min_child_weight': np.arange(1, 10),
    'xgbregressor__max_depth': np.arange(5, 8),
    'xgbregressor__learning_rate': np.logspace(-1, 0, 100),
    'xgbregressor__gamma': np.linspace(0, 0.3, num=10),
    'xgbregressor__colsample_bytree': np.linspace(0.4, 1, 10),
}
grid_X=random_search(XGBRegressor(),preprocessor,param_grid_X,30,5)
test_score_param(grid_X)
```

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

```
from xgboost import XGBRegressor
param_grid_X={
    'xgbregressor__subsample': np.linspace(0.9, 1.0, 10),
    'xgbregressor__reg_lambda': np.linspace(0.7,0.9, num=10),
    'xgbregressor__reg_alpha': np.arange(40,70,10),
    'xgbregressor__n_estimators': np.arange(800, 850, 10),
    'xgbregressor__min_child_weight': np.arange(1, 10),
    'xgbregressor__max_depth': np.arange(6, 7),
    # 'xgbregressor__learning_rate': np.logspace(-1, 0, 100),
    'xgbregressor__learning_rate': np.linspace(0.15,0.5,num=100),
    'xgbregressor__gamma': np.linspace(0, 0.2, num=10),
    'xgbregressor__colsample_bytree': np.linspace(0.7, 1, 10),
}
grid_X=random_search(XGBRegressor(),preprocessor,param_grid_X,30,5)
test_score_param(grid_X)
```

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

```
param_grid_X={
    'xgbregressor__subsample': [1],
    'xgbregressor__reg_lambda':[0.8] ,
    'xgbregressor__reg_alpha': [50],
    'xgbregressor__n_estimators': [880],
    'xgbregressor__min_child_weight': [2],
    'xgbregressor__max_depth': [7],
    'xgbregressor__learning_rate': [0.41],
    'xgbregressor__gamma': [0.05],
    'xgbregressor__colsample_bytree': [0.95]
}
# grid_X=random_search(XGBRegressor(),preprocessor,param_random_X,30,5)
grid_X=grid_search(XGBRegressor(),preprocessor,param_grid_X)
test_score_param(grid_X)
```

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：

```
param_grid = {
        "gradientboostingregressor__n_estimators": [50,200,500],
        "gradientboostingregressor__max_depth": [3,5,7],
        "gradientboostingregressor__learning_rate": [0.001,0.01,0.1,1],
        "gradientboostingregressor__subsample": [0.3,0.5,0.9],
    }
# grid_G=random_search(GradientBoostingRegressor(),preprocessor,param_random,10,5)
grid_G=grid_search(GradientBoostingRegressor(),preprocessor,param_grid,cv)
test_score_param(grid_G)
```

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：

```
from sklearn.tree import DecisionTreeRegressor

param_grid_D={
#     "decisiontreeregressor__splitter":["best","random"],
    'decisiontreeregressor__max_depth': np.arange(3, 10),
     'decisiontreeregressor__min_samples_leaf': np.arange(1, 10),
     "decisiontreeregressor__min_weight_fraction_leaf":np.linspace(0.0, 0.5, 5),
#     'decisiontreeregressor__max_features': ['auto', 'sqrt', 'log2'],
     'decisiontreeregressor__max_leaf_nodes':[None,10,20,30,40,50,60,70,80,90]
}
grid_D=random_search(DecisionTreeRegressor(),preprocessor,param_grid_D,100,5)
# grid_D=grid_search(DecisionTreeRegressor(),preprocessor,param_grid_D)
test_score_param(grid_D)
```

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：

```
from sklearn.tree import DecisionTreeRegressor

param_grid_D={
    'decisiontreeregressor__min_samples_leaf':np.arange(1,5),
    "decisiontreeregressor__min_weight_fraction_leaf": [0.0],
    'decisiontreeregressor__max_leaf_nodes':[None]
}
# grid_D=random_search(DecisionTreeRegressor(),preprocessor,param_grid_D,50,5)
grid_D=grid_search(DecisionTreeRegressor(),preprocessor,param_grid_D)
test_score_param(grid_D)
```

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