Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Технологии машинного обучения»

Отчет по лабораторной работе №6:

«Ансамбли моделей машинного обучения»

Выполнил:
студент группы ИУ5-62
Андреев Артем

Подпись и дата:

Проверил:

Подпись и дата:

Москва, 2019 г.

In [1]:
```
import numpy as np
import pandas as pd
pd.set_option('display.max.rows', 1000)
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style='ticks')
```

Подготовка датасета

In [2]:
```
# House Sales in King County, USA
# Predict house price using regression
data = pd.read_csv('data/kc_house_data.csv')
data.shape
```

Out[2]:
```
(21613, 21)
```

In [3]:
```
data.head()
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basem
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	

5 rows × 21 columns

In [4]:
```
data = data.drop(columns=['id', 'date', ])
data.corr()
```

Out[4]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_bas
price	1.000000	0.308350	0.525138	0.702035	0.089661	0.256794	0.266369	0.397293	0.036362	0.667434	0.605567	0.3
bedrooms	0.308350	1.000000	0.515884	0.576671	0.031703	0.175429	-0.006582	0.079532	0.028472	0.356967	0.477600	0.3
bathrooms	0.525138	0.515884	1.000000	0.754665	0.087740	0.500653	0.063744	0.187737	-0.124982	0.664983	0.685342	0.2
sqft_living	0.702035	0.576671	0.754665	1.000000	0.172826	0.353949	0.103818	0.284611	-0.058753	0.762704	0.876597	0.4
sqft_lot	0.089661	0.031703	0.087740	0.172826	1.000000	-0.005201	0.021604	0.074710	-0.008958	0.113621	0.183512	0.0
floors	0.256794	0.175429	0.500653	0.353949	-0.005201	1.000000	0.023698	0.029444	-0.263768	0.458183	0.523885	-0.2
waterfront	0.266369	-0.006582	0.063744	0.103818	0.021604	0.023698	1.000000	0.401857	0.016653	0.082775	0.072075	0.0
view	0.397293	0.079532	0.187737	0.284611	0.074710	0.029444	0.401857	1.000000	0.045990	0.251321	0.167649	0.2
condition	0.036362	0.028472	-0.124982	-0.058753	-0.008958	-0.263768	0.016653	0.045990	1.000000	-0.144674	-0.158214	0.1
grade	0.667434	0.356967	0.664983	0.762704	0.113621	0.458183	0.082775	0.251321	-0.144674	1.000000	0.755923	0.1
sqft_above	0.605567	0.477600	0.685342	0.876597	0.183512	0.523885	0.072075	0.167649	-0.158214	0.755923	1.000000	-0.0
sqft_basement	0.323816	0.303093	0.283770	0.435043	0.015286	-0.245705	0.080588	0.276947	0.174105	0.168392	-0.051943	1.0
yr_built	0.054012	0.154178	0.506019	0.318049	0.053080	0.489319	-0.026161	-0.053440	-0.361417	0.446963	0.423898	-0.1
yr_renovated	0.126434	0.018841	0.050739	0.055363	0.007644	0.006338	0.092885	0.103917	-0.060618	0.014414	0.023285	0.0
zipcode	-0.053203	-0.152668	-0.203866	-0.199430	-0.129574	-0.059121	0.030285	0.084827	0.003026	-0.184862	-0.261190	0.0
lat	0.307003	-0.008931	0.024573	0.052529	-0.085683	0.049614	-0.014274	0.006157	-0.014941	0.114084	-0.000816	0.1
long	0.021626	0.129473	0.223042	0.240223	0.229521	0.125419	-0.041910	-0.078400	-0.106500	0.198372	0.343803	-0.1
sqft_living15	0.585379	0.391638	0.568634	0.756420	0.144608	0.279885	0.086463	0.280439	-0.092824	0.713202	0.731870	0.2
sqft_lot15	0.082447	0.029244	0.087175	0.183286	0.718557	-0.011269	0.030703	0.072575	-0.003406	0.119248	0.194050	0.0

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

In [6]:
```
data.dtypes
```

Out[6]:
```
price            float64
bedrooms           int64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
grade              int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

In [7]:
```
# пропусков нет, разделим на обучающую и тестовую выборку
from sklearn.model_selection import train_test_split
```

In [8]:
```
# перед этим разделим исходный датасет на 2: один содержит независимые параметры, другой — зависимый (price)
X, y = data[data.columns[range(1, 19)]], data[data.columns[[0]]]
```

In [9]:
```
X.dtypes
```

Out[9]:
```
bedrooms           int64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
grade              int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

In [10]:
```
y.dtypes
```

Out[10]:
```
price    float64
dtype: object
```

In [11]:
```
test_size = 0.2
state = 42
xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size=test_size, random_state=state)
len(xTrain), len(xTest), len(yTrain), len(yTest)
```

Out[11]:
```
(17290, 4323, 17290, 4323)
```

Обучение моделей

BaggingRegressor with DecisionTreeRegressor

In [12]:
```
from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeRegressor

state = 42

bagreg_treereg = BaggingRegressor(DecisionTreeRegressor(random_state=state), n_estimators=100)
bagreg_treereg.fit(X, y.values.ravel())
```

Out[12]:
```
BaggingRegressor(base_estimator=DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=42, splitter='best'),
         bootstrap=True, bootstrap_features=False, max_features=1.0,
         max_samples=1.0, n_estimators=100, n_jobs=None, oob_score=False,
         random_state=None, verbose=0, warm_start=False)
```

In [13]:
```
bagreg_treeregTrain = bagreg_treereg.predict(xTrain)
```

In [14]:
```
bagreg_treeregTest = bagreg_treereg.predict(xTest)
```

In [64]:
```
# оценим качество модели регрессии
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# 1) mean absolute error – средняя абсолютная ошибка
print('mean_absolute_error (train): {}'.format(mean_absolute_error(yTrain, bagreg_treeregTrain)))
print('mean_absolute_error  (test): {}'.format(mean_absolute_error(yTest, bagreg_treeregTest)))
```

```
mean_absolute_error (train): 25312.712859152005
mean_absolute_error  (test): 26819.885470475383
```

In [31]:
```
from xgboost import XGBRegressor

xgbreg_treereg = XGBRegressor(n_jobs=-1)
```

In [32]:
```
xgbreg_treereg.fit(X, y.values.ravel())
```

```
[23:06:53] WARNING: src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:square
derror.
```

Out[32]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0,
       importance_type='gain', learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=-1, nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1)
```

In [33]:
```
xgbreg_treeregTrain = xgbreg_treereg.predict(xTrain)
```

In [34]:
```
xgbreg_treeregTest = xgbreg_treereg.predict(xTest)
```

In [65]:
```
# оценим качество модели регрессии
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# 1) mean absolute error – средняя абсолютная ошибка
print('mean_absolute_error (train): {}'.format(mean_absolute_error(yTrain, xgbreg_treeregTrain)))
print('mean_absolute_error  (test): {}'.format(mean_absolute_error(yTest, xgbreg_treeregTest)))
```

```
mean_absolute_error (train): 72965.081767821
mean_absolute_error  (test): 75977.6707147814
```

подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации

In [41]:
```
from sklearn.model_selection import GridSearchCV, KFold, ShuffleSplit
parameters = [{'n_estimators': np.array(range(1, 101))}]
bagreg_grid = GridSearchCV(BaggingRegressor(DecisionTreeRegressor(random_state=state)), parameters, cv=Shuffl
eSplit(n_splits=10, test_size=0.2), scoring='neg_mean_absolute_error',
                           n_jobs=-1,
                           verbose=10,
                          )
bagreg_grid.fit(X, y.values.ravel())
```

```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    1.3s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:    1.5s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    2.4s
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    3.2s
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:    4.5s
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:    5.8s
[Parallel(n_jobs=-1)]: Done  69 tasks      | elapsed:    8.2s
[Parallel(n_jobs=-1)]: Done  82 tasks      | elapsed:   10.7s
[Parallel(n_jobs=-1)]: Done  97 tasks      | elapsed:   14.3s
[Parallel(n_jobs=-1)]: Done 112 tasks      | elapsed:   18.2s
[Parallel(n_jobs=-1)]: Done 129 tasks      | elapsed:   23.5s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:   29.1s
[Parallel(n_jobs=-1)]: Done 165 tasks      | elapsed:   36.8s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:   44.6s
[Parallel(n_jobs=-1)]: Done 205 tasks      | elapsed:   55.2s
[Parallel(n_jobs=-1)]: Done 226 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 249 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 272 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 297 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 322 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done 376 tasks      | elapsed:  2.9min
[Parallel(n_jobs=-1)]: Done 405 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done 434 tasks      | elapsed:  3.9min
[Parallel(n_jobs=-1)]: Done 465 tasks      | elapsed:  4.4min
[Parallel(n_jobs=-1)]: Done 496 tasks      | elapsed:  5.0min
[Parallel(n_jobs=-1)]: Done 529 tasks      | elapsed:  5.7min
[Parallel(n_jobs=-1)]: Done 562 tasks      | elapsed:  6.4min
[Parallel(n_jobs=-1)]: Done 597 tasks      | elapsed:  7.2min
[Parallel(n_jobs=-1)]: Done 632 tasks      | elapsed:  8.0min
[Parallel(n_jobs=-1)]: Done 669 tasks      | elapsed:  9.0min
[Parallel(n_jobs=-1)]: Done 706 tasks      | elapsed:  9.9min
[Parallel(n_jobs=-1)]: Done 745 tasks      | elapsed: 11.1min
[Parallel(n_jobs=-1)]: Done 784 tasks      | elapsed: 12.2min
[Parallel(n_jobs=-1)]: Done 825 tasks      | elapsed: 13.6min
[Parallel(n_jobs=-1)]: Done 866 tasks      | elapsed: 14.9min
[Parallel(n_jobs=-1)]: Done 909 tasks      | elapsed: 16.5min
[Parallel(n_jobs=-1)]: Done 952 tasks      | elapsed: 18.1min
[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed: 19.9min finished
```

Out[41]:
```
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=None, test_size=0.2, train_size=None),
       error_score='raise-deprecating',
       estimator=BaggingRegressor(base_estimator=DecisionTreeRegressor(criterion='mse', max_depth=None, max_
features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
       ...stimators=10, n_jobs=None, oob_score=False,
         random_state=None, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid=[{'n_estimators': array([  1,   2, ...,  99, 100])}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='neg_mean_absolute_error', verbose=10)
```

In [42]:
```
bagreg_grid.best_estimator_
```

Out[42]:
```
BaggingRegressor(base_estimator=DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=42, splitter='best'),
         bootstrap=True, bootstrap_features=False, max_features=1.0,
         max_samples=1.0, n_estimators=88, n_jobs=None, oob_score=False,
         random_state=None, verbose=0, warm_start=False)
```

In [44]:
```
bagreg_grid.best_score_
```

Out[44]:
```
-69083.26590519112
```

In [45]:
```
bagreg_grid.best_params_
```

Out[45]:
```
{'n_estimators': 88}
```

In [49]:
```
parameters = [{"colsample_bytree": [1.0], "min_child_weight": [0.8, 1.0, 1.2],
               'max_depth': range(3, 11), 'n_estimators': [25, 50, 75, 100]}]
xgbreg_grid = GridSearchCV(XGBRegressor(), parameters, cv=ShuffleSplit(n_splits=10, test_size=0.2), scoring=
'neg_mean_absolute_error',
                           n_jobs=-1,
                           verbose=10,
                          )
xgbreg_grid.fit(X, y.values.ravel())
```

```
Fitting 10 folds for each of 96 candidates, totalling 960 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    1.8s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:    2.5s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:    3.2s
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    5.8s
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    8.4s
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:   10.4s
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:   11.8s
[Parallel(n_jobs=-1)]: Done  69 tasks      | elapsed:   15.6s
[Parallel(n_jobs=-1)]: Done  82 tasks      | elapsed:   17.9s
[Parallel(n_jobs=-1)]: Done  97 tasks      | elapsed:   20.4s
[Parallel(n_jobs=-1)]: Done 112 tasks      | elapsed:   24.7s
[Parallel(n_jobs=-1)]: Done 129 tasks      | elapsed:   27.5s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:   31.8s
[Parallel(n_jobs=-1)]: Done 165 tasks      | elapsed:   37.9s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:   42.4s
[Parallel(n_jobs=-1)]: Done 205 tasks      | elapsed:   48.8s
[Parallel(n_jobs=-1)]: Done 226 tasks      | elapsed:   53.7s
[Parallel(n_jobs=-1)]: Done 249 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 272 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 297 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 322 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 376 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 405 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done 434 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done 465 tasks      | elapsed:  2.3min
[Parallel(n_jobs=-1)]: Done 496 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done 529 tasks      | elapsed:  2.8min
[Parallel(n_jobs=-1)]: Done 562 tasks      | elapsed:  3.1min
[Parallel(n_jobs=-1)]: Done 597 tasks      | elapsed:  3.3min
[Parallel(n_jobs=-1)]: Done 632 tasks      | elapsed:  3.6min
[Parallel(n_jobs=-1)]: Done 669 tasks      | elapsed:  3.9min
[Parallel(n_jobs=-1)]: Done 706 tasks      | elapsed:  4.2min
[Parallel(n_jobs=-1)]: Done 745 tasks      | elapsed:  4.6min
[Parallel(n_jobs=-1)]: Done 784 tasks      | elapsed:  4.9min
[Parallel(n_jobs=-1)]: Done 825 tasks      | elapsed:  5.3min
[Parallel(n_jobs=-1)]: Done 866 tasks      | elapsed:  5.7min
[Parallel(n_jobs=-1)]: Done 909 tasks      | elapsed:  6.3min
[Parallel(n_jobs=-1)]: Done 960 out of 960 | elapsed:  6.8min finished
```

```
[00:03:01] WARNING: src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:square
derror.
```

Out[49]:
```
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=None, test_size=0.2, train_size=None),
       error_score='raise-deprecating',
       estimator=XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0,
       importance_type='gain', learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid=[{'colsample_bytree': [1.0], 'min_child_weight': [0.8, 1.0, 1.2], 'max_depth': range(3, 1
1), 'n_estimators': [25, 50, 75, 100]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='neg_mean_absolute_error', verbose=10)
```

In [50]:
```
xgbreg_grid.best_estimator_
```

Out[50]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1.0, gamma=0,
       importance_type='gain', learning_rate=0.1, max_delta_step=0,
       max_depth=9, min_child_weight=0.8, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1)
```

In [51]:
```
xgbreg_grid.best_score_
```

Out[51]:
```
-65791.59962085068
```

In [52]:
```
xgbreg_grid.best_params_
```

Out[52]:
```
{'colsample_bytree': 1.0,
 'max_depth': 9,
 'min_child_weight': 0.8,
 'n_estimators': 100}
```

Обучение с новыми параметрами

In [62]:
```
bagreg_grid.best_estimator_.fit(xTrain, yTrain.values.ravel())

bagreg_treeregTrainNew = bagreg_grid.best_estimator_.predict(xTrain)
bagreg_treeregTestNew = bagreg_grid.best_estimator_.predict(xTest)

print('mean_absolute_error (train): {}'.format(mean_absolute_error(yTrain, bagreg_treeregTrain)))
print('mean_absolute_error  (test): {}'.format(mean_absolute_error(yTest, bagreg_treeregTest)))

print('New mean_absolute_error (train): {}'.format(mean_absolute_error(yTrain, bagreg_treeregTrainNew)))
print('New mean_absolute_error  (test): {}'.format(mean_absolute_error(yTest, bagreg_treeregTestNew)))
```

```
mean_absolute_error (train): 25312.712859152005
mean_absolute_error  (test): 26819.885470475383
New mean_absolute_error (train): 26035.191542553624
New mean_absolute_error  (test): 73352.7241413403
```

In [63]:
```
xgbreg_grid.best_estimator_.fit(xTrain, yTrain.values.ravel())

xgbreg_treeregTrainNew = xgbreg_grid.best_estimator_.predict(xTrain)
xgbreg_treeregTestNew = xgbreg_grid.best_estimator_.predict(xTest)

print('mean_absolute_error (train): {}'.format(mean_absolute_error(yTrain, xgbreg_treeregTrain)))
print('mean_absolute_error  (test): {}'.format(mean_absolute_error(yTest, xgbreg_treeregTest)))

print('New mean_absolute_error (train): {}'.format(mean_absolute_error(yTrain, xgbreg_treeregTrainNew)))
print('New mean_absolute_error  (test): {}'.format(mean_absolute_error(yTest, xgbreg_treeregTestNew)))
```

```
[00:14:27] WARNING: src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:square
derror.
mean_absolute_error (train): 72965.081767821
mean_absolute_error  (test): 75977.6707147814
New mean_absolute_error (train): 33665.23249439705
New mean_absolute_error  (test): 69095.68306876012
```

Улучшение для xgbreg_grid

In [ ]: