Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Домашнее задание
по дисциплине
«Методы машинного обучения»

Выполнил:
Хотин П.Ю.
ИУ5-24М

Москва, 2020 год

Задача

В ходе выполнения проекта необходимо решить задачу регрессии, обучив алгоритм предсказывать данные на существующем датасете.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

В качестве набора данных будем использовать датасет с платформы Kaggle. Будем использовать регрессию для прогноза цена на недвижимость. Датасет содержит данные о широте и долготе местоположения дома, площадь комнат и дома в целом, близость к океану и др.

Будем решать задачу регрессии, предсказывая стоимость дома.

Посмотрим, что представлено в данных

```
data_path = "./housing.csv"
data = pd.read_csv(data_path)

data.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Видно, что столбец ocean_proximity имеет тип object. Преобразуем его:

```
new_val = pd.get_dummies(data.ocean_proximity)
```

```
data[new_val.columns] = new_val
```

```
data.describe()
```

ooms	total_bedrooms	population	households	median_income	median_house_value	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN
00000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
53081	537.870553	1425.476744	499.539680	3.870671	206855.816909	0.442636	0.317393	0.000242	0.110950	0.128779
15252	421.385070	1132.462122	382.329753	1.899822	115395.615874	0.496710	0.465473	0.015563	0.314077	0.334963
00000	1.000000	3.000000	1.000000	0.499900	14999.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50000	296.000000	787.000000	280.000000	2.563400	119600.000000	0.000000	0.000000	0.000000	0.000000	0.000000
00000	435.000000	1166.000000	409.000000	3.534800	179700.000000	0.000000	0.000000	0.000000	0.000000	0.000000
00000	647.000000	1725.000000	605.000000	4.743250	264725.000000	1.000000	1.000000	0.000000	0.000000	0.000000
00000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
 10  <1H OCEAN           20640 non-null  uint8  
 11  INLAND              20640 non-null  uint8  
 12  ISLAND              20640 non-null  uint8  
 13  NEAR BAY            20640 non-null  uint8  
 14  NEAR OCEAN          20640 non-null  uint8  
dtypes: float64(9), object(1), uint8(5)
memory usage: 1.7+ MB
```

```
data = data[['longitude', 'latitude', 'housing_median_age',
'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income'
       , '<1H OCEAN', 'INLAND',
       'ISLAND', 'NEAR BAY', 'NEAR OCEAN','median_house_value']]
```

Корреляционный анализ, выбор подходящих признаков

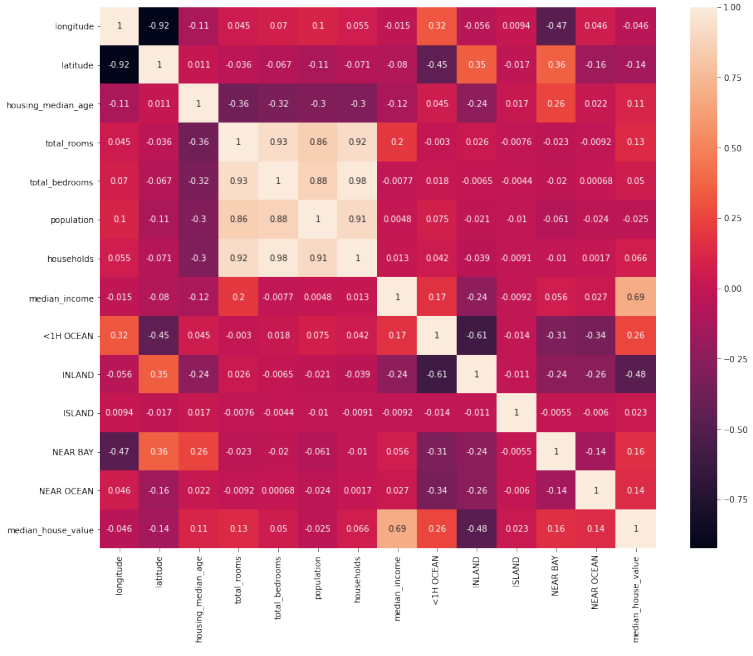
```
corr = data.corr()
```

```
corr
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	<1H OCEAN	INLAND	I
longitude	1.000000	-0.924664	-0.108197	0.044568	0.069608	0.099773	0.055310	-0.015176	0.321121	-0.055575	0
latitude	-0.924664	1.000000	0.011173	-0.036100	-0.066983	-0.108785	-0.071035	-0.079809	-0.446969	0.351166	-0
housing_median_age	-0.108197	0.011173	1.000000	-0.361262	-0.320451	-0.296244	-0.302916	-0.119034	0.045300	-0.236645	0
total_rooms	0.044568	-0.036100	-0.361262	1.000000	0.930380	0.857126	0.918484	0.198050	-0.003031	0.025624	-0
total_bedrooms	0.069608	-0.066983	-0.320451	0.930380	1.000000	0.877747	0.979728	-0.007723	0.018314	-0.006463	-0
population	0.099773	-0.108785	-0.296244	0.857126	0.877747	1.000000	0.907222	0.004834	0.074613	-0.020732	-0
households	0.055310	-0.071035	-0.302916	0.918484	0.979728	0.907222	1.000000	0.013033	0.042435	-0.039402	-0
median_income	-0.015176	-0.079809	-0.119034	0.198050	-0.007723	0.004834	0.013033	1.000000	0.168876	-0.237496	-0
<1H OCEAN	0.321121	-0.446969	0.045300	-0.003031	0.018314	0.074613	0.042435	0.168876	1.000000	-0.607669	-0
INLAND	-0.055575	0.351166	-0.236645	0.025624	-0.006463	-0.020732	-0.039402	-0.237496	-0.607669	1.000000	-0
ISLAND	0.009446	-0.016572	0.017020	-0.007572	-0.004361	-0.010412	-0.009077	-0.009228	-0.013872	-0.010614	1
NEAR BAY	-0.474489	0.358771	0.255172	-0.023022	-0.019873	-0.060880	-0.010093	0.056197	-0.314813	-0.240887	-0
NEAR OCEAN	0.045509	-0.160818	0.021622	-0.009175	0.000679	-0.024264	0.001714	0.027344	-0.342620	-0.262163	-0
median_house_value	-0.045967	-0.144160	0.105623	0.134153	0.049686	-0.024650	0.065843	0.688075	0.256617	-0.484859	0

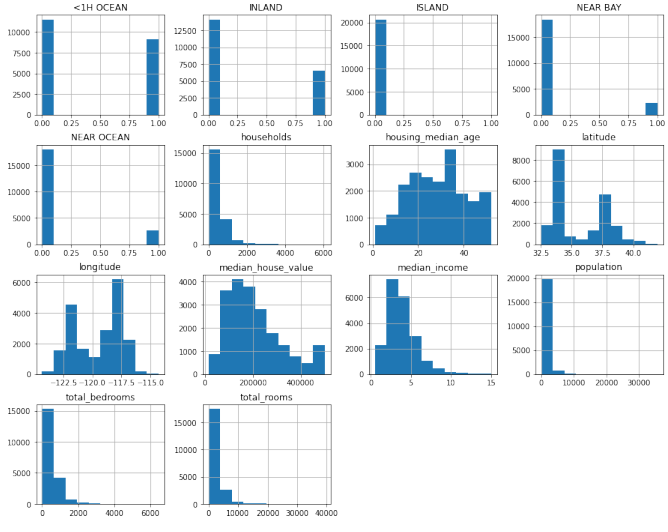
```
plt.figure(figsize=(15,12))
sns.heatmap(corr, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x108731e10>
```



```
data.hist(figsize=(15,12))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x124983150>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x10878f790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1087c5e10>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x123fc14d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x123ff6b50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x124c84210>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x125082910>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x1250b8ed0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1250b8f10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1250fb6d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x125173310>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x1251aa990>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1251dffd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1252206d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x125257d50>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x12529b410>]],
      dtype=object)
```



```
data.isna().sum()
```

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
<1H OCEAN               0
INLAND                  0
ISLAND                  0
NEAR BAY                0
NEAR OCEAN              0
median_house_value      0
dtype: int64
```

Видны нулевые строки, их нужно удалить.

```
data = data.fillna(data.mean())
```

```
data.isna().sum()
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
<1H OCEAN             0
INLAND                0
ISLAND                0
NEAR BAY              0
NEAR OCEAN            0
```

```
median_house_value    0
dtype: int64
```

```
from sklearn import preprocessing
convert = preprocessing.StandardScaler()
```

```
label = data['median_house_value']
```

```
label
```

```
0        452600.0
1        358500.0
2        352100.0
3        341300.0
4        342200.0
           ...
20635     78100.0
20636     77100.0
20637     92300.0
20638     84700.0
20639     89400.0
Name: median_house_value, Length: 20640, dtype: float64
```

Выделим целевой признак.

```
data = data.drop(['median_house_value'], axis=1)
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
```

```
from sklearn.linear_model import  LassoCV, Lasso
from sklearn.model_selection import  GridSearchCV
from sklearn.ensemble import RandomForestRegressor, BaggingRegressor
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    data, label, test_size=0.2, random_state=1)
```

```
X_train.shape, y_train.shape
```

```
((16512, 13), (16512,))
```

```
X_test.shape, y_test.shape
```

```
((4128, 13), (4128,))
```

```
def quality(test, predicted):
    print("  Метрики качества:")
    print("     Средняя квадратичная ошибка: "+
str(mean_squared_error(test, predicted)))
    print("     Средняя абсолютная ошибка: "+
str(mean_absolute_error(test, predicted)))
    print("     Коэффициент детерминации: "+str(r2_score(test,
predicted)))
```

Выбор моделей¶

В качестве моделей регрессии выберем модель Lasso, а также две ансамблевые модели: BaggingRegressor и RandomForestRegressor

```
models = [Lasso(), BaggingRegressor(), RandomForestRegressor()]
models
```

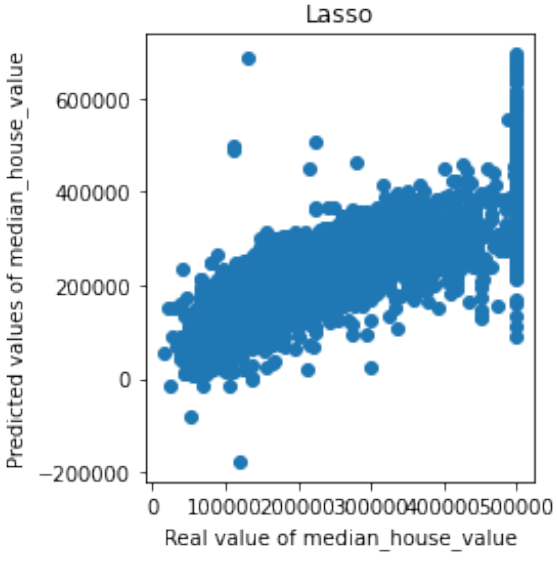
```
[Lasso(), BaggingRegressor(), RandomForestRegressor()]
```

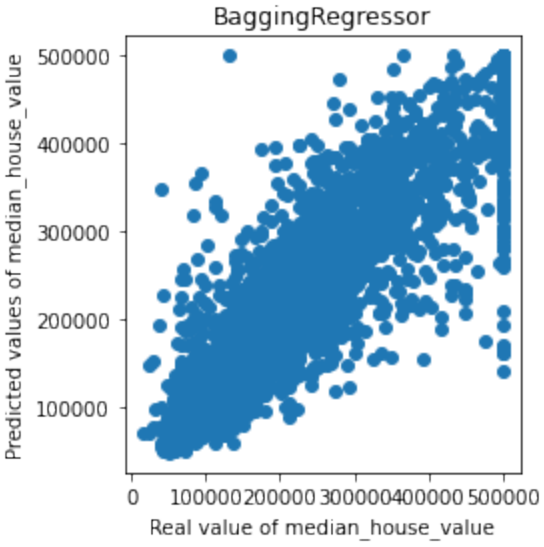
Базовое решение для всех моделей

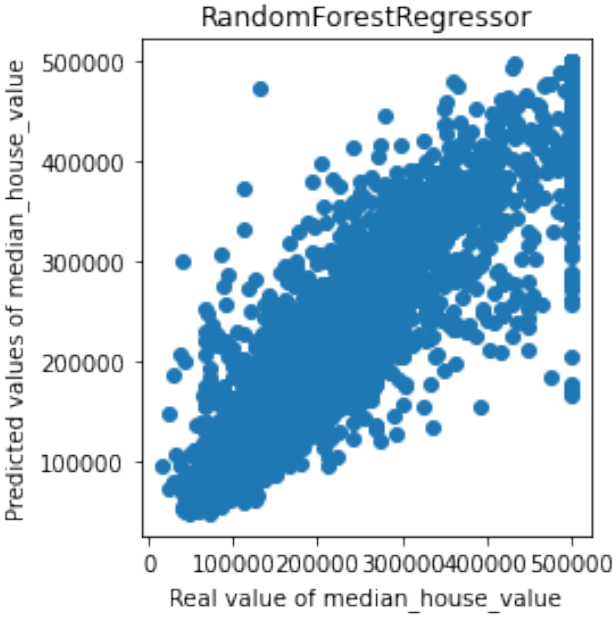
```
for model in models:
    print("Обучение модели "+type(model).__name__)
    model.fit(X_train, y_train)
    predicted = model.predict(X_test)
    plt.figure(figsize=(4, 4))
    plt.scatter(y_test,predicted)
    plt.title(type(model).__name__)
    plt.xlabel('Real value of median_house_value')
    plt.ylabel('Predicted values of median_house_value')
    plt.tight_layout()
    quality(y_test, predicted)
```

```
Обучение модели Lasso
```

```
  Метрики качества:
     Средняя квадратичная ошибка: 4754024388.947851
     Средняя абсолютная ошибка: 49767.04192499784
     Коэффициент детерминации: 0.637565176864489
Обучение модели BaggingRegressor
  Метрики качества:
     Средняя квадратичная ошибка: 2643311868.994971
     Средняя абсолютная ошибка: 33295.50794573643
     Коэффициент детерминации: 0.7984805732258311
Обучение модели RandomForestRegressor
  Метрики качества:
     Средняя квадратичная ошибка: 2392491194.1802464
     Средняя абсолютная ошибка: 31714.124677810076
     Коэффициент детерминации: 0.8176025085542538
```




BaggingRegressor
Predicted values of median_house_value
500000
400000
300000
200000
100000
0
100000
200000
300000
400000
500000
Real value of median_house_value


RandomForestRegressor
Predicted values of median_house_value
500000
400000
300000
200000
100000
0
100000
200000
300000
400000
500000
Real value of median_house_value

Подбор гиперпараметров моделей

Подбор гиперпараметров для модели BaggingRegressor

```
params_bag_GS = {"n_estimators": [1,10,20],
                 "max_features":[1,2,4,6,8],
                 "max_samples": [0.5,0.1],
                "bootstrap": [True, False],
            "bootstrap_features": [True, False]}
Bag_model_GS = GridSearchCV(estimator=BaggingRegressor(),
param_grid=params_bag_GS)
Bag_model_GS.fit(X_train,y_train)
print(Bag_model_GS)
print(Bag_model_GS.best_score_)
print(Bag_model_GS.best_estimator_)
```

```
GridSearchCV(estimator=BaggingRegressor(),
             param_grid={'bootstrap': [True, False],
                         'bootstrap_features': [True, False],
                         'max_features': [1, 2, 4, 6, 8],
                         'max_samples': [0.5, 0.1],
                         'n_estimators': [1, 10, 20]})
0.7861684015117929
BaggingRegressor(max_features=8, max_samples=0.5, n_estimators=20)
```

Подбор гиперпараметров для модели RandomForestRegressor

```
params_RF = {"max_depth": [3,5,8,9],
                 "max_features":['auto', 'sqrt', 'log2'],
                 "min_samples_split": [2, 3,5,7],
                 "min_samples_leaf": [1, 3,5,6]}
model_RF_GS = GridSearchCV(RandomForestRegressor(),
param_grid=params_RF)
model_RF_GS.fit(X_train,y_train)
print(model_RF_GS)
print(model_RF_GS.best_score_)
print(model_RF_GS.best_estimator_)
```

```
GridSearchCV(estimator=RandomForestRegressor(),
             param_grid={'max_depth': [3, 5, 8, 9],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 3, 5, 6],
                         'min_samples_split': [2, 3, 5, 7]})
0.7718500187637947
RandomForestRegressor(max_depth=9, min_samples_leaf=3,
min_samples_split=7)
```

Подбор гиперпараметров для модели Lasso

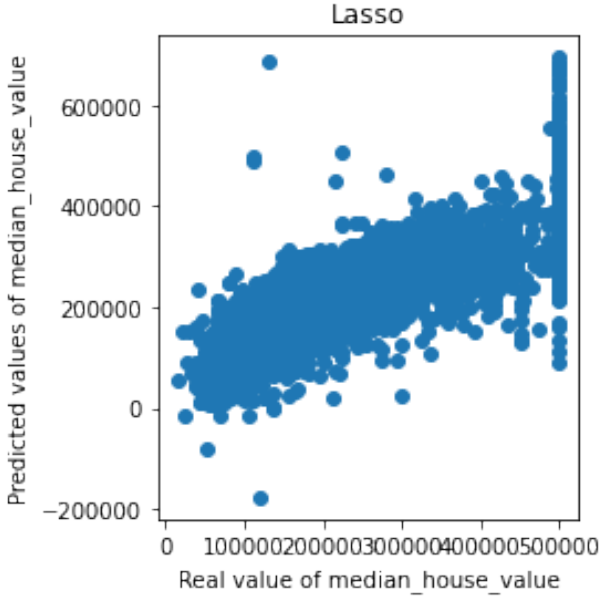
```
param_grid = { 'alpha': [i/100 for i in range(1,5)]}
model_lasso = GridSearchCV(estimator=Lasso(), param_grid=param_grid,
n_jobs=-1)
model_lasso.fit(X_train, y_train)
print(model_lasso)
print(model_lasso.best_score_)
print(model_lasso.best_estimator_)
```

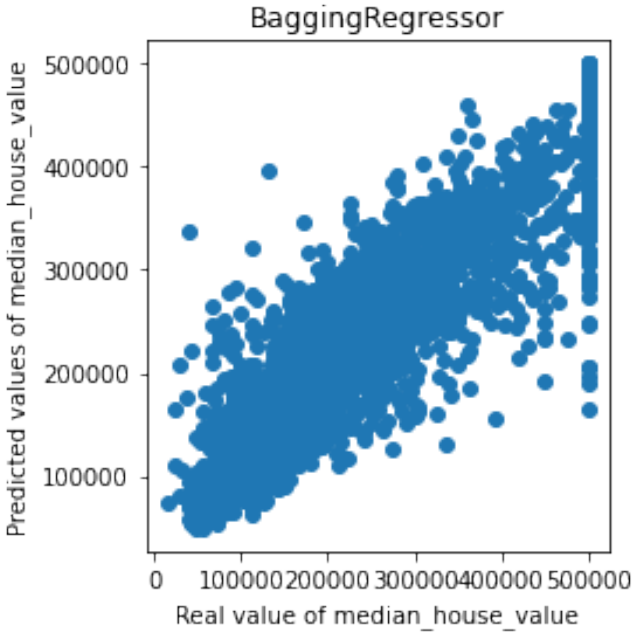
```
GridSearchCV(estimator=Lasso(), n_jobs=-1,
             param_grid={'alpha': [0.01, 0.02, 0.03, 0.04]})
0.6447618177420328
Lasso(alpha=0.04)
```

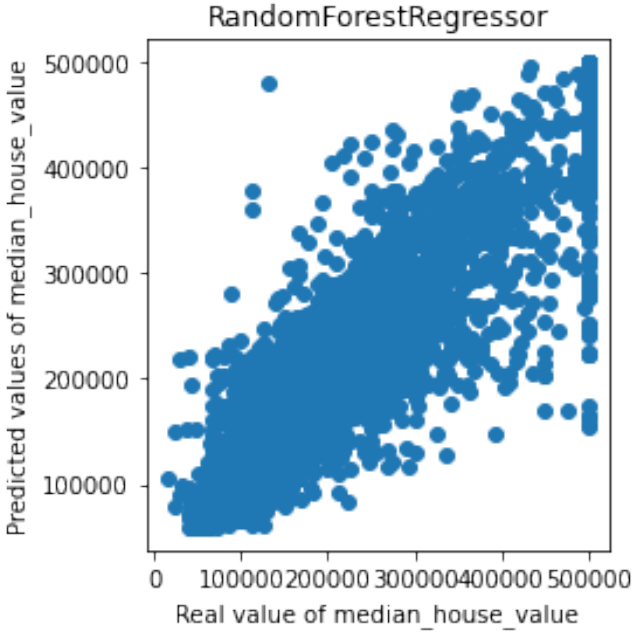
```
models = [Lasso(alpha=0.04),
    BaggingRegressor(max_features=8, max_samples=0.5,
n_estimators=20),
          RandomForestRegressor(max_depth=9, min_samples_leaf=3,
min_samples_split=7)
         ]

for model in models:
    print("Обучение модели "+type(model).__name__)
    model.fit(X_train, y_train)
    predicted = model.predict(X_test)
    plt.figure(figsize=(4, 4))
```

```
    plt.scatter(y_test,predicted)
    plt.title(type(model).__name__)
    plt.xlabel('Real value of median_house_value')
    plt.ylabel('Predicted values of median_house_value')
    plt.tight_layout()
    quality(y_test, predicted)
```

```
Обучение модели Lasso


  Метрики качества:
     Средняя квадратичная ошибка: 4754049665.262504
     Средняя абсолютная ошибка: 49766.777312948994
     Коэффициент детерминации: 0.637563249862042
Обучение модели BaggingRegressor
  Метрики качества:
     Средняя квадратичная ошибка: 2475750508.079202
     Средняя абсолютная ошибка: 33846.92794331395
     Коэффициент детерминации: 0.8112550285586724
Обучение модели RandomForestRegressor
  Метрики качества:
     Средняя квадратичная ошибка: 3000890094.411342
     Средняя абсолютная ошибка: 37451.83523656864
     Коэффициент детерминации: 0.77121971163093
```







Вывод

В результате работы с моделями ансамблевые модели показали лучший результат как с гиперпаметрами, так и без них.