Рубежный контроль №2

Мирсонов Вячеслав Александрович

Группа РТ5-61Б

Вариант 9

Постановка задачи

Условие:

Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Методы:

Метод №1: "Дерево решений". Метод №2: "Градиентный бустинг".

Набор данных: Houses to rent in Brazil.

https://www.kaggle.com/rubenssjr/brasilian-houses-to-rent

Импорт библиотек:

In [36]:

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

Загрузка и первичная подготовка данных:

In [38]:

```
# Загрузка данных
data = pd.read_csv("houses_to_rent_v2.csv")
data.shape
```

Out[38]:

```
(499, 13)
```

In [39]:

```
data
```

Out[39]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa (R$)	rent amount (R$)	property tax (R$)	fire insurance (R$)	total (R$)
0	São Paulo	70	2	1	1	7	acept	furnished	2065	3300	211	42	5618
1	São Paulo	320	4	4	0	20	acept	not furnished	1200	4960	1750	63	7973
2	Porto Alegre	80	1	1	1	6	acept	not furnished	1000	2800	0	41	3841
3	Porto Alegre	51	2	1	0	2	acept	not furnished	270	1112	22	17	1421
4	São Paulo	25	1	1	0	1	not acept	not furnished	0	800	25	11	836
...	...	...	...	...	...	...	...	...	...	...	...	...	...
494	São Paulo	120	3	3	2	9	acept	furnished	1200	3500	109	45	4854
495	Rio de Janeiro	90	2	2	0	5	acept	not furnished	830	2500	84	33	3447
496	Porto Alegre	125	4	2	0	7	acept	not furnished	1006	1632	150	14	2802
497	São Paulo	200	3	4	3	6	acept	furnished	1800	5058	1200	65	8123
498	São Paulo	190	3	3	2	5	acept	not furnished	1800	3260	292	42	5394

499 rows × 13 columns

In [41]:

```
# уникальные значения столбца 'city'
data['city'].unique()
```

Out[41]:

```
array(['São Paulo', 'Porto Alegre', 'Rio de Janeiro', 'Campinas',
       'Belo Horizonte'], dtype=object)
```

In [42]:

```
# список колонок с типами данных
data.dtypes
```

Out[42]:

```
city                  object
area                   int64
rooms                  int64
bathroom               int64
parking spaces         int64
floor                  int64
animal                object
furniture             object
hoa (R$)               int64
rent amount (R$)       int64
property tax (R$)      int64
fire insurance (R$)    int64
total (R$)             int64
dtype: object
```

Обработка пропусков в данных:

In [43]:

```
# проверим, есть ли пропущенные значения
data.isnull().sum()
```

Out[43]:

```
city                  0
area                  0
rooms                 0
bathroom              0
parking spaces        0
floor                 0
animal                0
furniture             0
hoa (R$)              0
rent amount (R$)      0
property tax (R$)     0
fire insurance (R$)   0
total (R$)            0
dtype: int64
```

Пропущенных данных нет, можем присутпать к масштабированию данных

Масштабирование данных:

MinMax масштабирование:

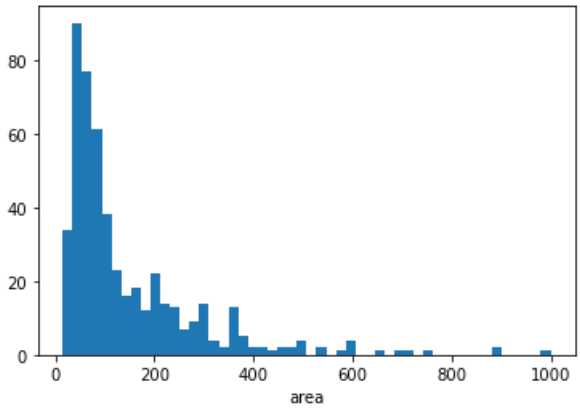
$$ x_{новый} = \frac{x_{старый} - min(X)}{max(X)-min(X)} $$

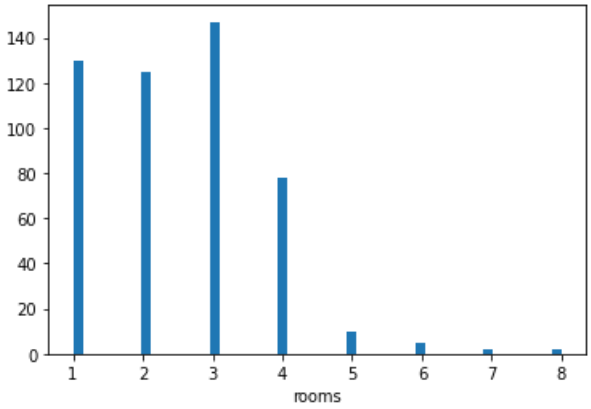
In [45]:

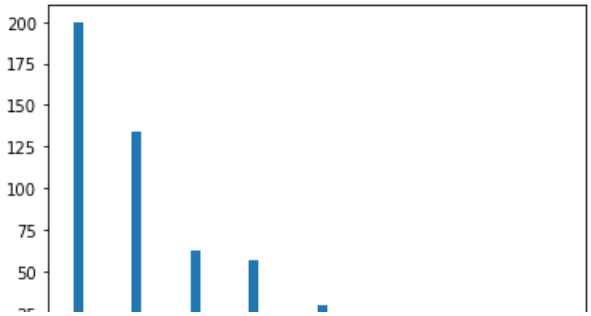
```
from sklearn.preprocessing import MinMaxScaler
```

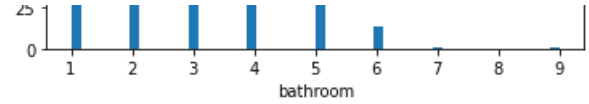
In [48]:

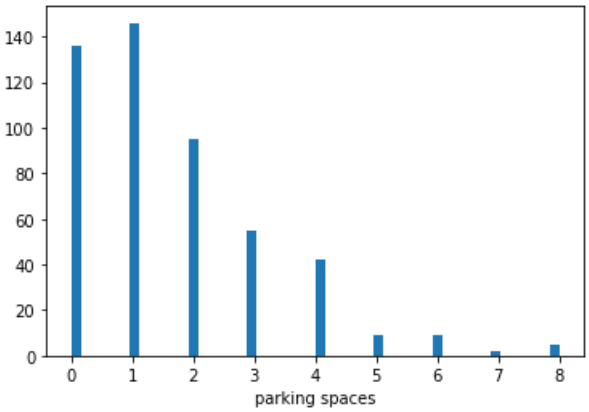
```
# список колонок с числовыми данными
num_cols = ['area', 'rooms', 'bathroom', 'parking spaces', 'floor', 'hoa (R$)', 'rent amount (R$)','property t
# Гистограмма по признакам
for col in data[num_cols]:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```

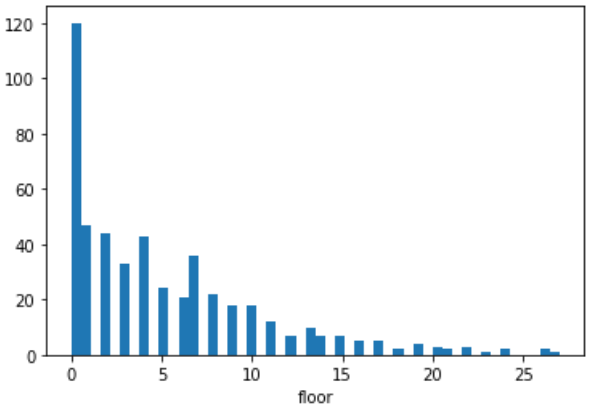


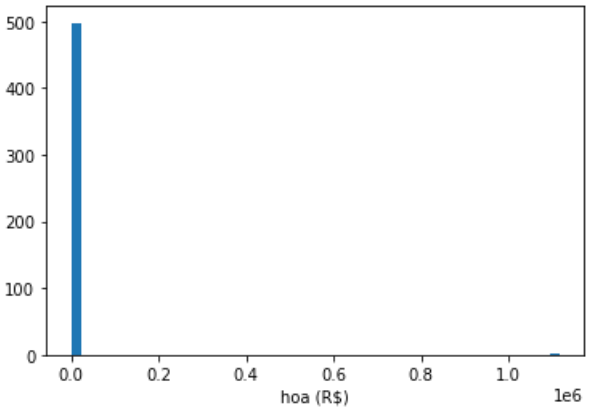


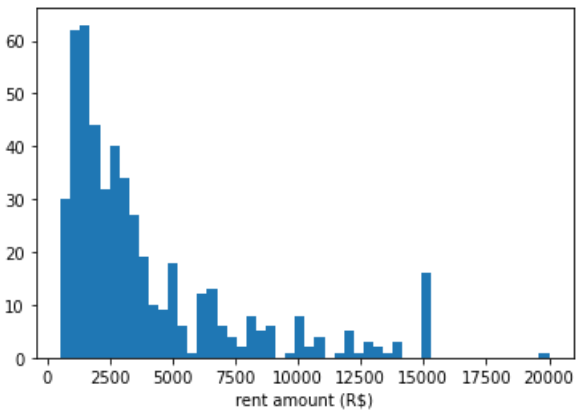



25
0
1
2
3
4
5
6
7
8
9
bathroom

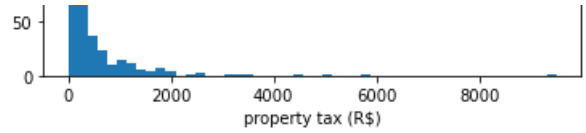

140
120
100
80
60
40
20
0
0
1
2
3
4
5
6
7
8
parking spaces

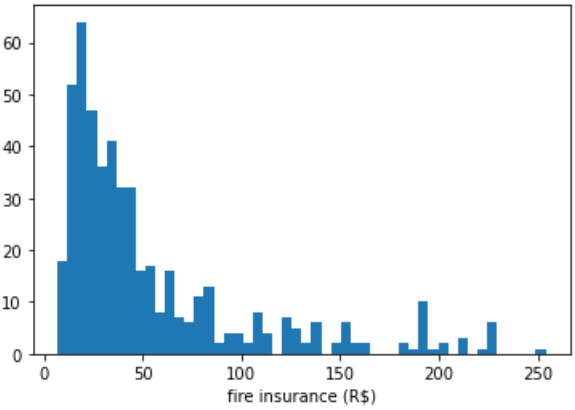

120
100
80
60
40
20
0
0
5
10
15
20
25
floor

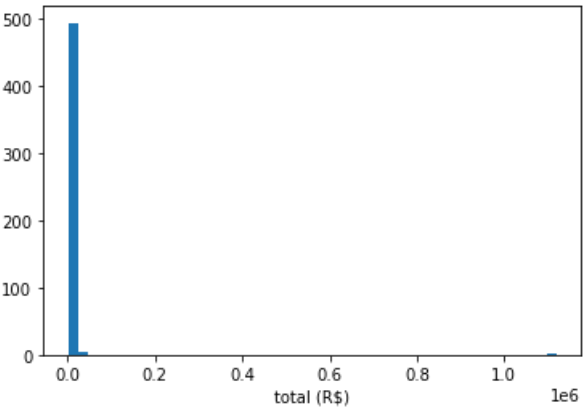

500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
hoa (R$)
1e6


60
50
40
30
20
10
0
0
2500
5000
7500
10000
12500
15000
17500
20000
rent amount (R$)


300
250
200
150
100





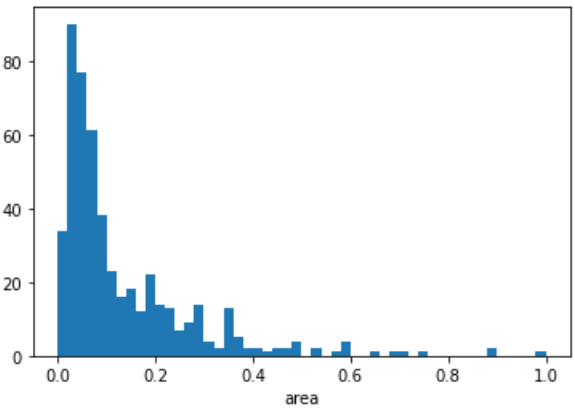


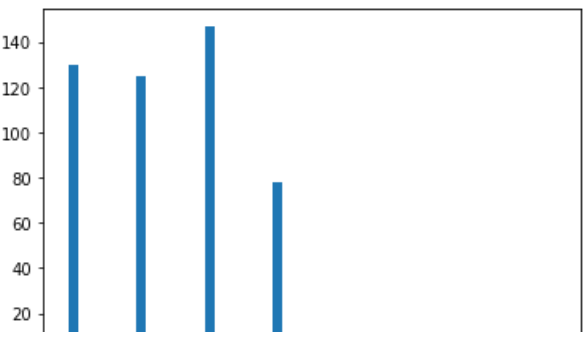
In [49]:

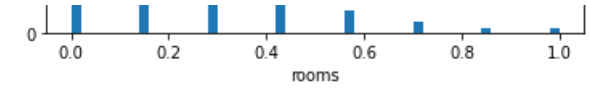
```
# MinMax масштабирование
sc1 = MinMaxScaler()
for item in num_cols:
    data.loc[:, item] = sc1.fit_transform(data[[item]])
```

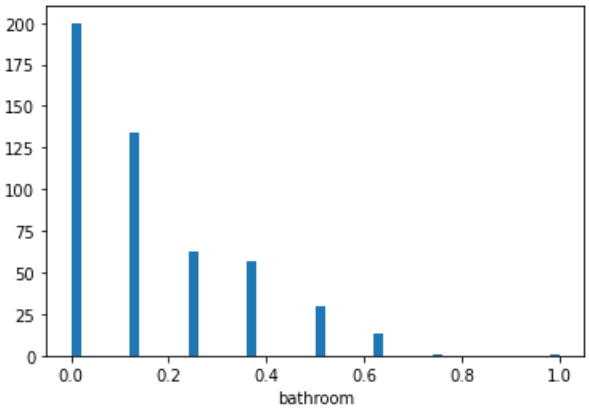
In [50]:

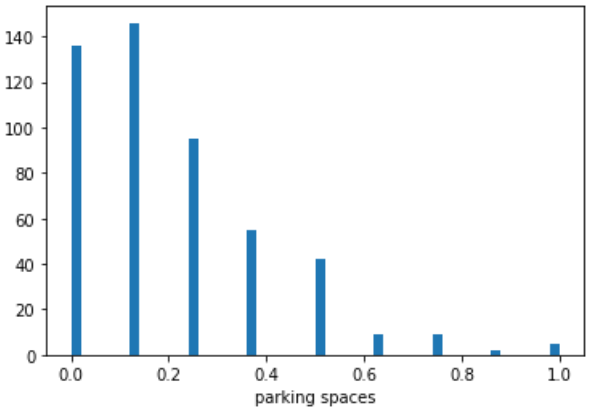
```
# Гистограмма по отмасштабированным признакам
for col in data[num_cols]:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```

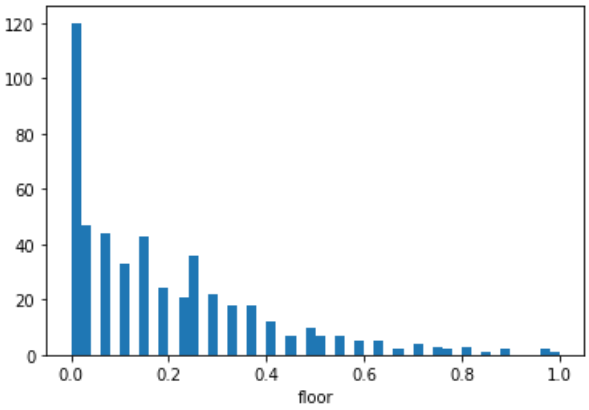


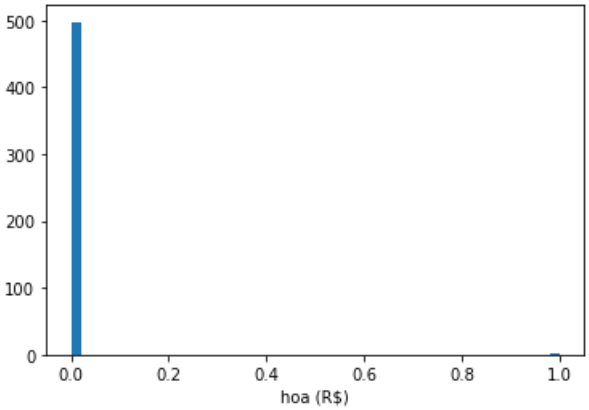


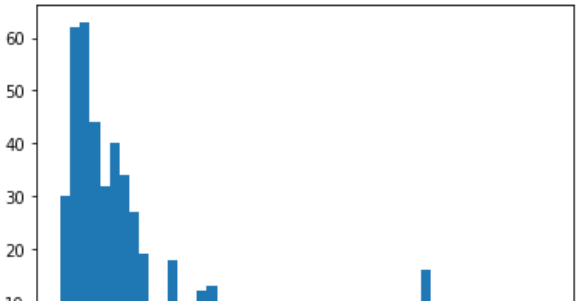

0
0.0
0.2
0.4
0.6
0.8
1.0
rooms

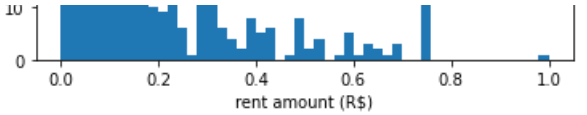

200
175
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0
bathroom

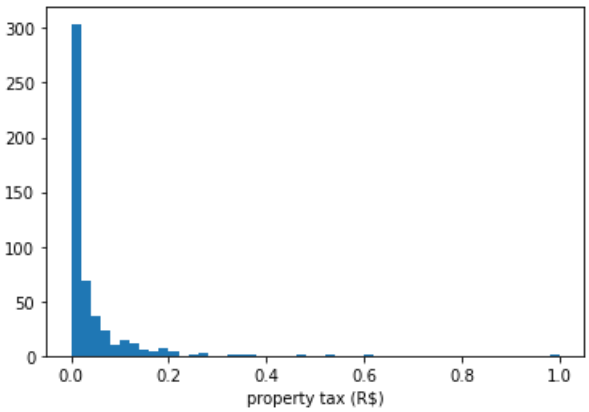

140
120
100
80
60
40
20
0
0.0
0.2
0.4
0.6
0.8
1.0
parking spaces

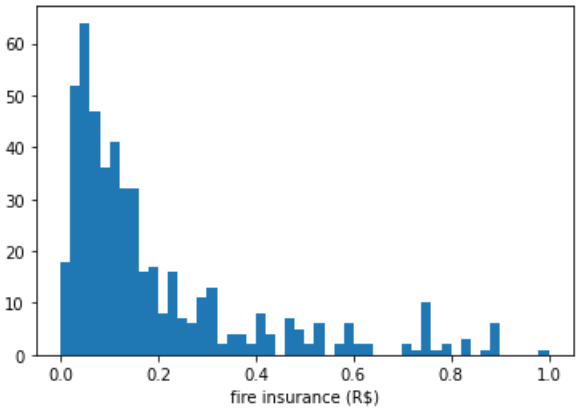

120
100
80
60
40
20
0
0.0
0.2
0.4
0.6
0.8
1.0
floor

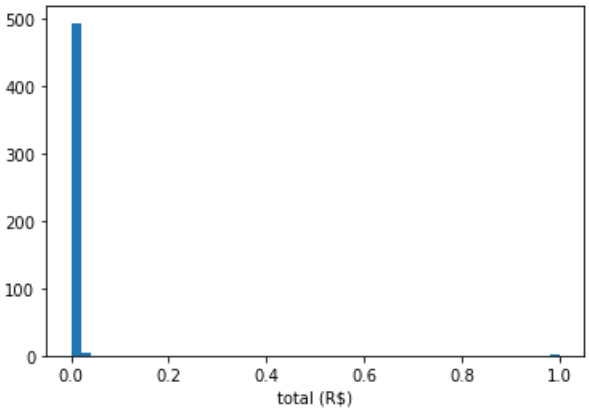

500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
hoa (R$)


60
50
40
30
20









Кодирование категориальных признаков:

In [51]:

```
# Выберем категориальные колонки
# Цикл по колонкам датасета
for col in data.columns:
    dt = str(data[col].dtype)
    if dt=='object':
        temp_un = data[col].nunique()
        print('Колонка {}. Тип данных {}. Количество уникальных значений {}.'.format(col, dt, temp_un))
```

```
Колонка city. Тип данных object. Количество уникальных значений 5.
Колонка animal. Тип данных object. Количество уникальных значений 2.
Колонка furniture. Тип данных object. Количество уникальных значений 2.
```

Кодирование категорий наборами бинарных значений - one-hot encoding

In [52]:

```
cat_cols = ['city', 'animal', 'furniture']
one_hot = pd.get_dummies(data[cat_cols].astype(str))
one_hot.head()
```

Out[52]:

	city_Belo Horizonte	city_Campinas	city_Porto Alegre	city_Rio de Janeiro	city_São Paulo	animal_acept	animal_not acept	furniture_furnished	furniture_not furnished
0	0	0	0	0	1	1	0	1	0
1	0	0	0	0	1	1	0	0	1
2	0	0	1	0	0	1	0	0	1
3	0	0	1	0	0	1	0	0	1
4	0	0	0	0	1	0	1	0	1

In [53]:

```
# Замена исходных категориальных колонок наборами бинарных значений
data = data.join(one_hot)
data.drop(columns=cat_cols, inplace=True)
```

In [54]:

```
# первые 5 строк получившегося набора данных
data.head()
```

Out[54]:

	area	rooms	bathroom	parking spaces	floor	hoa (R$)	rent amount (R$)	property tax (R$)	fire insurance (R$)	total (R$)	city_Belo Horizonte	city_Campinas	city_Porto Alegre	city_Rio de Janeiro
0	0.055894	0.142857	0.000	0.125	0.259259	0.001849	0.141388	0.022211	0.141700	0.004473	0	0	0	0
1	0.309959	0.428571	0.375	0.000	0.740741	0.001074	0.226735	0.184211	0.226721	0.006577	0	0	0	0
2	0.066057	0.000000	0.000	0.125	0.222222	0.000895	0.115681	0.000000	0.137652	0.002886	0	0	1	0
3	0.036585	0.142857	0.000	0.000	0.074074	0.000242	0.028895	0.002316	0.040486	0.000724	0	0	1	0
4	0.010163	0.000000	0.000	0.000	0.037037	0.000000	0.012853	0.002632	0.016194	0.000201	0	0	0	0

Построение моделей:

Разделение выборки на обучающую и тестовую

In [55]:

```
from sklearn.model_selection import train_test_split
data_train, data_test, data_y_train, data_y_test = train_test_split(data[data.columns.drop('total (R$)')], dat
```

Модель "Дерево решений"

In [56]:

```
from sklearn.tree import DecisionTreeRegressor
dtc = DecisionTreeRegressor(random_state=1).fit(data_train, data_y_train)
data_test_predicted_dtc = dtc.predict(data_test)
```

Модель "Градиентный бустинг"

In [57]:

```
from sklearn.ensemble import GradientBoostingRegressor

gboostreg = GradientBoostingRegressor(random_state=10).fit(data_train, data_y_train)
gboostreg_predict = gboostreg.predict(data_test)
```

Оценка качества моделей:

В качестве метрик для оценки качества моделей будем использовать Mean squared error (средняя квадратичная ошибка), как наиболее часто используемую метрику для оценки качества регрессии, и метрику R^2 (коэффициент детерминации), потому что эта метрика является нормированной.

In [58]:

```
from sklearn.metrics import mean_squared_error, r2_score
# Mean squared error - средняя квадратичная ошибка
print('Метрика MSE:\nДерево решений: {}\nГрадиентный бустинг: {}'.format(mean_squared_error(data_y_test, data_
```

```
Метрика MSE:
Дерево решений: 4.009125408728235e-07
Градиентный бустинг: 6.742776208219548e-07
```

In [60]:

```
#Метрика R2 или коэффициент детерминации
```

```
print('Метрика R\u00B2:\nДерево решений: {}\nГрадиентный бустинг: {}'.format(r2_score(data_y_test, data_test_p
```

```
Метрика R²:
Дерево решений: 0.974242110132498
Градиентный бустинг: 0.9566789089225258
```

Выводы о качестве построенных моделей:

Исходя из оценки качества построенных моделей, можно увидеть, что модель "Дерево решений" лучше справляется с задачей по сравнению с моделью "Градиентный бустинг", что может свидетельствовать о переобучении модели "Градиентный бустинг".