Лабораторная работа №4.

Ковалёва Елена. 20152.

In [1]:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import norm
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math
import pandas_profiling as pp
```

In [2]:

```
from sklearn.metrics import mean_absolute_error
```

In [3]:

```
houses = pd.read_csv('data.csv')
##загружаем Dataset и смотрим его значения.
```

In [4]:

```
houses.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           4600 non-null   object 
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64  
 5   sqft_lot       4600 non-null   int64  
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64  
 8   view           4600 non-null   int64  
 9   condition      4600 non-null   int64  
 10  sqft_above     4600 non-null   int64  
 11  sqft_basement  4600 non-null   int64  
 12  yr_built       4600 non-null   int64  
 13  yr_renovated   4600 non-null   int64  
 14  street         4600 non-null   object 
 15  city           4600 non-null   object 
 16  statezip       4600 non-null   object 
 17  country        4600 non-null   object 
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

In [5]:

```
houses.shape
```

Out[5]:

```
(4600, 18)
```

В нашем Dataset левая часть показывает количество домов-4600. В правой колонке набор данных-18. Пропущенных значений-0.

In [6]:

```
houses.head()
```

Out[6]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	co
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	

In [7]:

```
#Столбцы street,data,country-удалить,т.к. не имеют важного значения для цены на дома.
houses=houses.drop(['date'], axis = 1)
houses=houses.drop(['country'], axis = 1)
houses=houses.drop(['street'],axis =1)
houses.head(5)
```

Out[7]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	s
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	

In [8]:

```
print('Длина до обработки:',len(houses))
count, list_drop = 0, []
for i in range(len(houses)):
    if houses['price'][i] == 0:
        houses.drop(i, inplace = True)
        count += 1

print('Пропущенные значения:', count)
print('Длина после обработки:',len(houses))
```

```
Длина до обработки: 4600
Пропущенные значения: 49
Длина после обработки: 4551
```

In [9]:

```
##Теперь мы всё загружаем в одну колонку с помощью One-hot-encode.
houses_endcode=pd.get_dummies(houses)
houses_endcode.head(4)
```

Out[9]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	s
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	

4 rows × 134 columns

In [10]:

```
houses.shape
```

Out[10]:

```
(4551, 15)
```

In [11]:

```
X = houses.iloc[:, 1:]
X.shape
```

Out[11]:

```
(4551, 14)
```

In [12]:

```
#После того, как получили нужные нам данные. Нам остается их разбить на контрольную и обуча
houses.shape
X =np.array(houses_endcode)[:, 1:]
Y =np.array(houses_endcode)[:,0]
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=.2,shuffle=False)
```

In [13]:

```
model= sm.OLS(Y_train,X_train)
ols=model.fit()

y_test_ols=ols.predict(X_test)
rmse=0#RMSE-среднеквадратичное отклонение нужно для вывода ошибок предсказаний
for i in range(len(Y_test)):
    rmse+=(Y_test[i]-y_test_ols[i])**2
rmse=math.sqrt(rmse/len(Y_test))
print('Ошибка предсказаний =',rmse)
```

```
Ошибка предсказаний = 991125.3911514686
```

2.Исследуем значимость модели и коэффициентов линейной регрессии.

In [14]:

```
houses = sm.OLS(Y_train, X_train)
res = houses.fit()
print(res.summary())
#С помощью метода наименьших квадратов выводим нашу обучающую выборку для оценки параметров
##неизвестной модели линейной регрессии.
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                      y   R-squared:
0.808
Model:                            OLS   Adj. R-squared:
0.802
Method:                 Least Squares   F-statistic:                    1
41.2
Date:                Fri, 11 Dec 2020   Prob (F-statistic):
0.00
Time:                        15:56:14   Log-Likelihood:                -48
793.
No. Observations:                3640   AIC:                         9.780
e+04
Df Residuals:                    3534   BIC:                         9.845
e+04
Df Model:                         105
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
x1         -3.049e+04   4019.714     -7.584      0.000   -3.84e+04   -2.26
e+04
x2          1.938e+04   6389.819      3.033      0.002    6852.414    3.19
e+04
x3           140.0420      3.751     37.336      0.000     132.688     14
7.396
x4            -0.0247      0.081     -0.304      0.761      -0.183
0.134
x5         -4.466e+04   7997.176     -5.585      0.000   -6.03e+04    -2.9
e+04
x6          6.458e+05   3.66e+04     17.647      0.000    5.74e+05    7.18
e+05
x7          5.875e+04   4203.897     13.976      0.000    5.05e+04     6.7
e+04
x8          2.843e+04   4996.747      5.690      0.000    1.86e+04    3.82
e+04
x9           128.1238      4.149     30.882      0.000     119.990     13
6.258
x10           11.9153      5.407      2.204      0.028       1.314      2
2.517
x11         -217.0348    152.150     -1.426      0.154    -515.346      8
1.277
x12            3.5142      3.228      1.089      0.276      -2.815
9.843
x13         1.486e+05   2.03e+05      0.732      0.464   -2.49e+05    5.47
e+05
x14         1.724e+05   1.93e+05      0.892      0.372   -2.06e+05    5.51
```

```
e+05
x15         8.672e+04   2.23e+05      0.389      0.697    -3.5e+05    5.24
e+05
x16         3.665e+05   1.71e+05      2.142      0.032     3.1e+04    7.02
e+05
x17         1.771e+05   1.58e+05      1.124      0.261   -1.32e+05    4.86
e+05
x18         2.123e+05   1.72e+05      1.236      0.217   -1.25e+05    5.49
e+05
x19         4.007e+05    2.4e+05      1.667      0.096   -7.05e+04    8.72
e+05
x20         1.826e+05   1.53e+05      1.190      0.234   -1.18e+05    4.83
e+05
x21          3.48e+05   1.79e+05      1.946      0.052   -2580.467    6.99
e+05
x22         1.894e+05   2.06e+05      0.919      0.358   -2.15e+05    5.94
e+05
x23         3.981e+05   2.48e+05      1.605      0.109   -8.83e+04    8.85
e+05
x24         1.651e+05   1.53e+05      1.078      0.281   -1.35e+05    4.65
e+05
x25         1.278e+05   1.54e+05      0.831      0.406   -1.74e+05    4.29
e+05
x26         1.998e+05   1.55e+05      1.291      0.197   -1.04e+05    5.03
e+05
x27         1.059e+05   1.98e+05      0.534      0.593   -2.83e+05    4.94
e+05
x28         2.495e+05   1.97e+05      1.269      0.205   -1.36e+05    6.35
e+05
x29         3.168e+05   2.16e+05      1.468      0.142   -1.06e+05     7.4
e+05
x30         1.649e+05   1.75e+05      0.945      0.345   -1.77e+05    5.07
e+05
x31         1.598e+05   2.03e+05      0.786      0.432   -2.39e+05    5.58
e+05
x32         2.768e+05   1.57e+05      1.761      0.078   -3.15e+04    5.85
e+05
x33         2.754e+05   2.41e+05      1.141      0.254   -1.98e+05    7.49
e+05
x34         1.343e+05   1.53e+05      0.878      0.380   -1.66e+05    4.34
e+05
x35         8.061e+05   1.55e+05      5.212      0.000    5.03e+05    1.11
e+06
x36         4.007e+05   1.52e+05      2.631      0.009    1.02e+05    6.99
e+05
x37         1.859e+05   1.74e+05      1.071      0.284   -1.55e+05    5.26
e+05
x38         3.079e+05    2.2e+05      1.402      0.161   -1.23e+05    7.38
e+05
x39          5.05e+05   2.47e+05      2.043      0.041    2.04e+04     9.9
e+05
x40         1.658e+05   1.53e+05      1.084      0.278   -1.34e+05    4.66
e+05
x41         1.337e+05   1.57e+05      0.851      0.395   -1.75e+05    4.42
e+05
x42         1.813e+05   1.62e+05      1.122      0.262   -1.35e+05    4.98
e+05
x43          1.49e+05   1.56e+05      0.955      0.340   -1.57e+05    4.55
e+05
x44         2.909e+05   2.21e+05      1.314      0.189   -1.43e+05    7.25
e+05
```

x45	1.919e+05	2.18e+05	0.882	0.378	-2.35e+05	6.18e+05
x46	3.072e+05	2.09e+05	1.467	0.142	-1.03e+05	7.18e+05
x47	4.063e+05	2.41e+05	1.683	0.092	-6.69e+04	8.8e+05
x48	4.153e+05	2.36e+05	1.761	0.078	-4.71e+04	8.78e+05
x49	3.223e+05	2.39e+05	1.351	0.177	-1.45e+05	7.9e+05
x50	1.232e+05	1.61e+05	0.765	0.444	-1.92e+05	4.39e+05
x51	1.551e+05	1.53e+05	1.013	0.311	-1.45e+05	4.55e+05
x52	2.497e+05	1.73e+05	1.445	0.148	-8.9e+04	5.88e+05
x53	3.816e+05	2.38e+05	1.602	0.109	-8.54e+04	8.49e+05
x54	9.49e+04	1.53e+05	0.621	0.535	-2.05e+05	3.94e+05
x55	2.713e+05	2.04e+05	1.332	0.183	-1.28e+05	6.71e+05
x56	6.584e+05	1.9e+05	3.468	0.001	2.86e+05	1.03e+06
x57	8.746e+04	1.2e+05	0.730	0.465	-1.47e+05	3.22e+05
x58	7.198e+04	1.22e+05	0.592	0.554	-1.67e+05	3.11e+05
x59	1.345e+05	1.27e+05	1.059	0.290	-1.15e+05	3.83e+05
x60	7.037e+05	1.37e+05	5.125	0.000	4.34e+05	9.73e+05
x61	2.323e+05	1.4e+05	1.662	0.097	-4.17e+04	5.06e+05
x62	1.946e+05	1.39e+05	1.403	0.161	-7.72e+04	4.66e+05
x63	1.676e+05	1.4e+05	1.197	0.231	-1.07e+05	4.42e+05
x64	1.614e+05	1.39e+05	1.160	0.246	-1.11e+05	4.34e+05
x65	1.771e+05	1.58e+05	1.124	0.261	-1.32e+05	4.86e+05
x66	1.651e+05	1.75e+05	0.942	0.346	-1.79e+05	5.09e+05
x67	1.826e+05	1.53e+05	1.190	0.234	-1.18e+05	4.83e+05
x68	1.651e+05	1.53e+05	1.078	0.281	-1.35e+05	4.65e+05
x69	1.278e+05	1.54e+05	0.831	0.406	-1.74e+05	4.29e+05
x70	9.281e+04	1.27e+05	0.733	0.463	-1.55e+05	3.41e+05
x71	1.998e+05	1.55e+05	1.291	0.197	-1.04e+05	5.03e+05
x72	8.967e+04	1.28e+05	0.701	0.483	-1.61e+05	3.41e+05
x73	2.121e+05	1.72e+05	1.233	0.218	-1.25e+05	5.49e+05
x74	1.774e+05	1.28e+05	1.385	0.166	-7.37e+04	4.28e+05
x75	1.01e+05	1.06e+05	0.956	0.339	-1.06e+05	3.08

```
e+05
x76         1.202e+05   1.05e+05      1.150      0.250   -8.48e+04    3.25
e+05
x77         5.607e+04   1.07e+05      0.526      0.599   -1.53e+05    2.65
e+05
x78         3.456e+05   1.59e+05      2.178      0.029    3.45e+04    6.57
e+05
x79         1.807e+05   1.59e+05      1.140      0.254    -1.3e+05    4.92
e+05
x80         1.343e+05   1.53e+05      0.878      0.380   -1.66e+05    4.34
e+05
x81         8.061e+05   1.55e+05      5.212      0.000    5.03e+05    1.11
e+06
x82         4.007e+05   1.52e+05      2.631      0.009    1.02e+05    6.99
e+05
x83          7.19e+04   1.04e+05      0.694      0.488   -1.31e+05    2.75
e+05
x84         1.658e+05   1.53e+05      1.084      0.278   -1.34e+05    4.66
e+05
x85         1.337e+05   1.57e+05      0.851      0.395   -1.75e+05    4.42
e+05
x86         1.813e+05   1.62e+05      1.122      0.262   -1.35e+05    4.98
e+05
x87          1.49e+05   1.56e+05      0.955      0.340   -1.57e+05    4.55
e+05
x88         2.228e+05   1.32e+05      1.694      0.090   -3.51e+04    4.81
e+05
x89         1.617e+05   1.33e+05      1.219      0.223   -9.84e+04    4.22
e+05
x90         9.493e+04   9.26e+04      1.025      0.306   -8.67e+04    2.77
e+05
x91         1.288e+05      9e+04      1.431      0.152   -4.76e+04    3.05
e+05
x92         5.168e+04   9.31e+04      0.555      0.579   -1.31e+05    2.34
e+05
x93         1.016e+05   8.97e+04      1.133      0.257   -7.43e+04    2.77
e+05
x94         1.228e+05   8.98e+04      1.367      0.172   -5.33e+04    2.99
e+05
x95         1.551e+05   1.53e+05      1.013      0.311   -1.45e+05    4.55
e+05
x96         2.497e+05   1.73e+05      1.445      0.148    -8.9e+04    5.88
e+05
x97          9.49e+04   1.53e+05      0.621      0.535   -2.05e+05    3.94
e+05
x98         1.423e+05   1.03e+05      1.380      0.168   -5.98e+04    3.44
e+05
x99         1.264e+05   1.14e+05      1.107      0.269   -9.76e+04    3.51
e+05
x100        1.369e+05   1.15e+05      1.190      0.234   -8.87e+04    3.63
e+05
x101         1.29e+05   1.03e+05      1.248      0.212   -7.37e+04    3.32
e+05
x102        4.016e+04   1.22e+05      0.328      0.743      -2e+05     2.8
e+05
x103        2.969e+05   7.51e+04      3.950      0.000     1.5e+05    4.44
e+05
x104         1.93e+05   6.67e+04      2.893      0.004    6.22e+04    3.24
e+05
x105        3.901e+05   6.92e+04      5.634      0.000    2.54e+05    5.26
e+05
```

```
x106       -2.226e+04   7.09e+04     -0.314      0.754   -1.61e+05    1.17
e+05
x107        2.054e+05   6.83e+04      3.008      0.003    7.15e+04    3.39
e+05
x108       -1.326e+04   7.29e+04     -0.182      0.856   -1.56e+05     1.3
e+05
x109        4.665e+05   7.06e+04      6.603      0.000    3.28e+05    6.05
e+05
x110        4.334e+05   6.76e+04      6.413      0.000    3.01e+05    5.66
e+05
x111        1.658e+05   6.78e+04      2.445      0.015    3.29e+04    2.99
e+05
x112         1.58e+05   6.95e+04      2.275      0.023    2.18e+04    2.94
e+05
x113        1.753e+05   6.78e+04      2.586      0.010    4.24e+04    3.08
e+05
x114         918.1183   6.93e+04      0.013      0.989   -1.35e+05    1.37
e+05
x115        3.881e+05   6.92e+04      5.611      0.000    2.52e+05    5.24
e+05
x116        2.043e+05   6.83e+04      2.993      0.003    7.05e+04    3.38
e+05
x117        1.472e+04   7.06e+04      0.208      0.835   -1.24e+05    1.53
e+05
x118        4.336e+04   7.03e+04      0.617      0.537   -9.44e+04    1.81
e+05
x119        4.582e+04   7.08e+04      0.647      0.518    -9.3e+04    1.85
e+05
x120        1.047e+05   7.02e+04      1.491      0.136    -3.3e+04    2.42
e+05
x121        1.485e+05   6.84e+04      2.171      0.030    1.44e+04    2.83
e+05
x122       -3.273e+04   7.07e+04     -0.463      0.643   -1.71e+05    1.06
e+05
x123       -6.806e+04   8.95e+04     -0.760      0.447   -2.44e+05    1.07
e+05
x124        8.355e+04   7.66e+04      1.091      0.275   -6.66e+04    2.34
e+05
x125       -1.338e+05   7.85e+04     -1.704      0.089   -2.88e+05    2.02
e+04
x126        -8.87e+04   7.13e+04     -1.245      0.213   -2.28e+05     5.1
e+04
x127        1.491e+05    7.3e+04      2.044      0.041    6052.493    2.92
e+05
x128       -1.394e+05   7.24e+04     -1.925      0.054   -2.81e+05    261
6.439
x129        -1.42e+05   8.72e+04     -1.628      0.104   -3.13e+05    2.91
e+04
x130       -1.706e+05   1.03e+05     -1.653      0.098   -3.73e+05    3.17
e+04
x131        2.479e+05   6.98e+04      3.549      0.000    1.11e+05    3.85
e+05
x132        1.232e+05   1.61e+05      0.765      0.444   -1.92e+05    4.39
e+05
x133        1.859e+05   1.74e+05      1.071      0.284   -1.55e+05    5.26
e+05
==============================================================================
====
Omnibus:                     3429.630   Durbin-Watson:
2.019
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           69155
```

```
7.818
Skew:                          3.922   Prob(JB):
0.00
Kurtosis:                     70.069   Cond. No.                     1.20
e+16
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
[2] The smallest eigenvalue is 4.19e-20. This might indicate that there ar
e
strong multicollinearity problems or that the design matrix is singular.
```

In [15]:

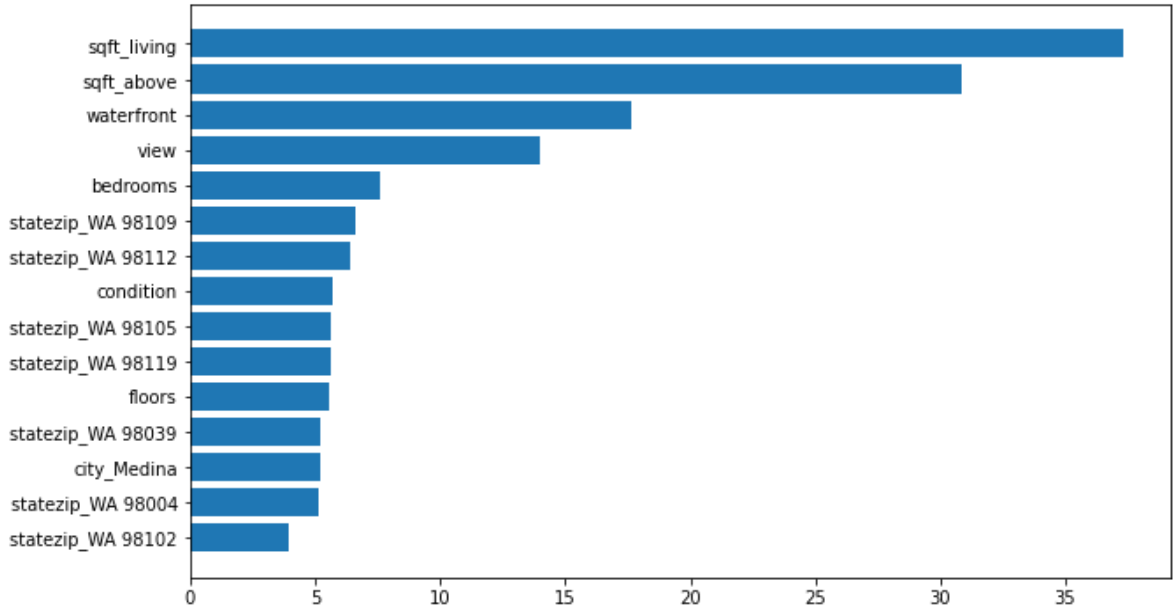
```
#Выводом P>|t|:
print(ols.pvalues)
y=ols.tvalues
x=[]
```

```
[4.24539098e-014 2.43869710e-003 1.75961925e-257 7.60877577e-001
 2.51290018e-008 7.36114269e-067 3.02498299e-043 1.37384414e-008
 1.31980499e-185 2.76132878e-002 1.53827088e-001 2.76393809e-001
 4.64039257e-001 3.72430488e-001 6.97225865e-001 3.22526839e-002
 2.61225157e-001 2.16633672e-001 9.55219548e-002 2.34211484e-001
 5.17094497e-002 3.58246709e-001 1.08656882e-001 2.81168665e-001
 4.05843708e-001 1.96725644e-001 5.93262296e-001 2.04691791e-001
 1.42243521e-001 3.44805714e-001 4.32049542e-001 7.84003929e-002
 2.53973252e-001 3.80013612e-001 1.97714388e-007 8.55530726e-003
 2.84455360e-001 1.61088405e-001 4.11207945e-002 2.78394246e-001
 3.95068764e-001 2.61799692e-001 3.39525880e-001 1.89091039e-001
 3.77828435e-001 1.42459394e-001 9.24070653e-002 7.83417363e-002
 1.76768895e-001 4.44292975e-001 3.11219548e-001 1.48461306e-001
 1.09221820e-001 5.34538839e-001 1.82851945e-001 5.30891795e-004
 4.65418729e-001 5.54175578e-001 2.89741148e-001 3.13100192e-007
 9.65345198e-002 1.60557409e-001 2.31282066e-001 2.46185918e-001
 2.61225157e-001 3.46311589e-001 2.34211484e-001 2.81168665e-001
 4.05843708e-001 4.63387446e-001 1.96725644e-001 4.83390914e-001
 2.17503618e-001 1.66091656e-001 3.38942952e-001 2.50364432e-001
 5.99078905e-001 2.94700654e-002 2.54470248e-001 3.80013612e-001
 1.97714388e-007 8.55530726e-003 4.87975049e-001 2.78394246e-001
 3.95068764e-001 2.61799692e-001 3.39525880e-001 9.04285030e-002
 2.22845407e-001 3.05596205e-001 1.52429372e-001 5.78746635e-001
 2.57409053e-001 1.71676829e-001 3.11219548e-001 1.48461306e-001
 5.34538839e-001 1.67595489e-001 2.68573347e-001 2.34110518e-001
 2.12149255e-001 7.42965781e-001 7.95753943e-005 3.83637708e-003
 1.90270072e-008 7.53616459e-001 2.64603932e-003 8.55576612e-001
 4.61680183e-011 1.61884742e-010 1.45186628e-002 2.29896786e-002
 9.75653060e-003 9.89431788e-001 2.16703855e-008 2.78151495e-003
 8.34862077e-001 5.37338072e-001 5.17512721e-001 1.35976805e-001
 3.00226324e-002 6.43490607e-001 4.47229450e-001 2.75459449e-001
 8.85595664e-002 2.13353471e-001 4.10685291e-002 5.43697805e-002
 1.03707961e-001 9.83674851e-002 3.92211252e-004 4.44292975e-001
 2.84455360e-001]
```

Заметим, что в нашей модели данных существуют (coef, std err, t, P>|t|,[0.025 0.975]). Рассмотрев колонку под P>|t|, то там коэффициент P>|t|=0.00==>эти переменные не влияют на цену дома.

In [16]:

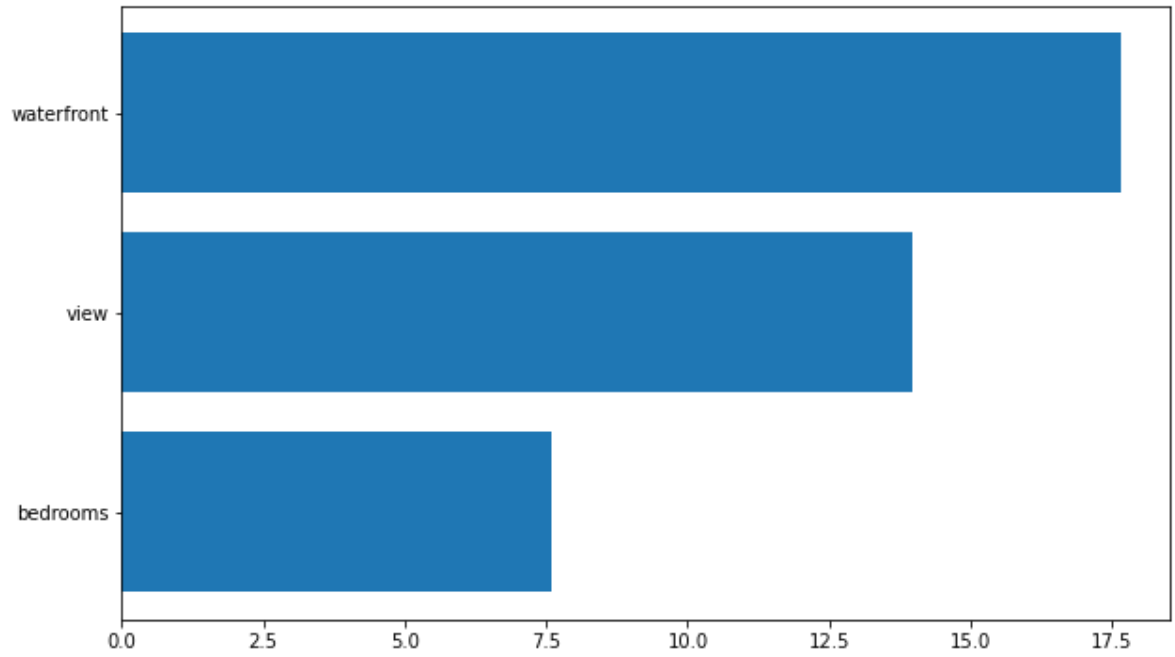
```
names=[]
params=[]

params=sorted(abs(ols.tvalues.copy()))[len(params)-15:]

for elem in params:
    index=list(abs(ols.tvalues)).index(elem)
    names.append((houses_endcode.columns[1:])[index])

plt.figure(figsize=(10,6))
plt.barh(names,params)
plt.show()
```



Самые важные признаки:sqft_living,sqft_above,waterfront,view,bedrooms.

In [17]:

```
#выводим waterfront,view и bedrooms:
names=[]
params=[]

params=sorted(abs(ols.tvalues.copy()))[len(params)-15:]
a=[params[10],params[11],params[12]]
for elem in a:
    index=list(abs(ols.tvalues)).index(elem)
    names.append((houses_endcode.columns[1:])[index])

plt.figure(figsize=(10,6))
plt.barh(names,a)
plt.show()
```



3. Строим решения методом бустинга. Сравниваем полученные результаты с линейной регрессией.

In [18]:

```
boost=GradientBoostingRegressor(random_state=0,n_estimators=10,max_depth=11)
boost.fit(X_train,Y_train)
y_test_boost=boost.predict(X_test)
```

In [19]:

```
rmse=0
for i in range(len(Y_test)):
    rmse+=(Y_test[i] - y_test_boost[i])**2
rmse=math.sqrt(rmse/len(Y_test))
print("Ошибка=",rmse)
```

```
Ошибка= 993768.8511497166
```

Ошибка линейной регрессии=1010994.6465697095 Ошибка бустинга=1003693.5702543714
Вывод:Градиентовый бустинг работает лучше.

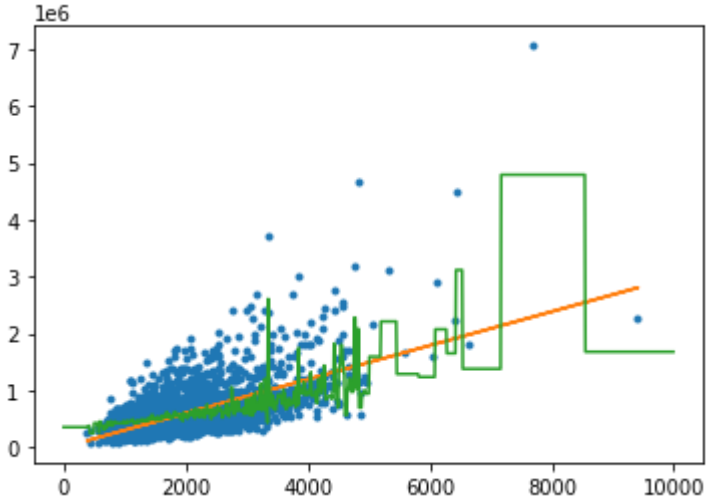
4. Строим графики полученных зависимостей для наиболее информативных переменных (отображаем на одном графике линейную регрессию и решение, полученное бустингом).

Строим график для переменной sqft_above

In [22]:

```
#plot_значение
x_range=X_train[:,8:9].copy()
plt.plot(x_range,Y_train, '.')

#plot_LinearReggression
model=sm.OLS(Y_train,x_range)
ols=model.fit()
ols_y=ols.predict(x_range)
plt.plot(x_range,ols_y)

y_test_ols=ols.predict(X_test[:,8:9])
rmse=0
for i in range(len(Y_test)):
    rmse+=(Y_test[i]-y_test_ols[i])**2
rmse=math.sqrt(rmse/len(Y_test))
print("Ошибка Линейной регрессии=",rmse)

#plot_GradientBossting
boost=GradientBoostingRegressor(random_state=0,n_estimators=10,max_depth=11)
boost.fit(x_range,Y_train)
y_boost=[]
x_arange= np.arange(0, 10000, 10)
for elem in x_arange:
    y_boost.append(boost.predict([[elem]]))
plt.plot(x_arange,y_boost)

y_test_boost=boost.predict(X_test[:,8:9])
rmse=0
for i in range(len(Y_test)):
    rmse+=(Y_test[i] - y_test_boost[i])**2
rmse=math.sqrt(rmse/len(Y_test))
print("Ошибка Градиентового бустинга =",rmse)
```

```
Ошибка Линейной регрессии= 1015050.8964889284
Ошибка Градиентового бустинга = 1027881.8223831767
```



Вывод:Если посмотреть график, то можно увидеть, что Линейная регрессия справляется хуже, чем Градиентовый бустинг. Это говорит о том, что Градиентовый бустинг выдает намного лучше точность.

In [ ]: