Московский государственный технический

университет им. Н. Э. Баумана

Отчёт по лабораторной работе №1 по курсу «Технологии машинного обучения».

«Разведочный анализ данных. Исследование и визуализация данных».

Выполнил:
Головацкий А. Д.
студент группы ИУ5-61Б

Проверил:
Гапанюк Ю. Е.

Подпись и дата:

Подпись и дата:

Москва, 2022 г.

ИУ5-61Б Головацкий Андрей

Разведочный анализ данных. Исследование и визуализация данных.

Оглавление

1) Текстовое описание набора данных (к оглавлению)

Датасет `real_estate_data.csv` содержит информацию об объявлениях о продаже квартир в Санкт-Петербурге.

Параметры:

- airports_nearest — расстояние до ближайшего аэропорта в метрах (м)
- balcony — число балконов
- ceiling_height — высота потолков (м)
- cityCenters_nearest — расстояние до центра города (м)
- days_exposition — сколько дней было размещено объявление (от публикации до снятия)
- first_day_exposition — дата публикации

- floor — этаж
- floors_total — всего этажей в доме
- is_apartment — апартаменты (булев тип)
- kitchen_area — площадь кухни в квадратных метрах ($м^2$)
- last_price — цена на момент снятия с публикации
- living_area — жилая площадь в квадратных метрах($м^2$)
- locality_name — название населённого пункта
- open_plan — свободная планировка (булев тип)
- parks_around3000 — число парков в радиусе 3 км
- parks_nearest — расстояние до ближайшего парка (м)
- ponds_around3000 — число водоёмов в радиусе 3 км
- ponds_nearest — расстояние до ближайшего водоёма (м)
- rooms — число комнат
- studio — квартира-студия (булев тип)
- total_area — площадь квартиры в квадратных метрах ($м^2$)
- total_images — число фотографий квартиры в объявлении

Пояснение: апартаменты — это нежилые помещения, не относящиеся к жилому фонду, но имеющие необходимые условия для проживания.

Формулирование гипотез (к оглавлению)

Целью анализа является подтверждение или опровержение определённых гипотез о тех или иных закономерностях в данных. Выдвинем 3 гипотезы.

Гипотеза №1:

Наибольшую часть объявлений составляют квартиры с площадью менее 60 м^2.

Гипотеза №2:

На стоимость квартиры сильнее всего влияет её площадь.

Гипотеза №3:

Радиус ценового "центра" Санкт-Петербурга составляет не более 8 км.

Подключение библиотек для анализа данных (к оглавлению)

```
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from datetime import datetime
import warnings
import numpy as np
import seaborn as sns
%matplotlib inline
sns.set(style="ticks")

warnings.filterwarnings('ignore')
```

Загрузка датасета из файла `real_estate_data.csv` (к оглавлению)

```
df = pd.read_csv('C:\\Users\\Andrew\\Anaconda Projects\\datasets\\
real_estate_data.csv', sep='\t')
```

2) Основные характеристики датасета (к оглавлению)

Выведем первые 5 строк датасета для проверки корректного импорта данных и изучения общей информации:

```
print(df.head(5))
```

```
   total_images  last_price  total_area first_day_exposition  rooms  \
0            20  13000000.0       108.0  2019-03-07T00:00:00      3
1             7   3350000.0        40.4  2018-12-04T00:00:00      1
2            10   5196000.0        56.0  2015-08-20T00:00:00      2
3             0  64900000.0       159.0  2015-07-24T00:00:00      3
4             2  10000000.0       100.0  2018-06-19T00:00:00      2

   ceiling_height  floors_total  living_area  floor is_apartment  ...
\
0            2.70          16.0         51.0      8          NaN  ...

1             NaN          11.0         18.6      1          NaN  ...

2             NaN           5.0         34.3      4          NaN  ...

3             NaN          14.0          NaN      9          NaN  ...

4            3.03          14.0         32.0     13          NaN  ...
```

```
   kitchen_area  balcony    locality_name  airports_nearest  \
0          25.0      NaN  Санкт-Петербург           18863.0
1          11.0      2.0   посёлок Шушары           12817.0
2           8.3      0.0  Санкт-Петербург           21741.0
3           NaN      0.0  Санкт-Петербург           28098.0
4          41.0      NaN  Санкт-Петербург           31856.0

  cityCenters_nearest  parks_around3000  parks_nearest
ponds_around3000  \
0             16028.0               1.0          482.0
2.0
1             18603.0               0.0            NaN
0.0
2             13933.0               1.0           90.0
2.0
3              6800.0               2.0           84.0
3.0
4              8098.0               2.0          112.0
1.0

   ponds_nearest  days_exposition
0          755.0              NaN
1            NaN             81.0
2          574.0            558.0
3          234.0            424.0
4           48.0            121.0

[5 rows x 22 columns]
```

Видим, что данные загружены корректно. Разбиения по строкам и столбцам произведены верно. Проблем с кодировкой не возникло.

Узнаем размер датасета:

```
print(f'Количество записей: {df.shape[0]}\nКоличество параметров:
{df.shape[1]}')
```

```
Количество записей: 23699
Количество параметров: 22
```

Посмотрим краткую информацию обо всех параматрах датасета:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23699 entries, 0 to 23698
Data columns (total 22 columns):
total_images            23699 non-null int64
last_price              23699 non-null float64
total_area              23699 non-null float64
```

```
first_day_exposition    23699 non-null object
rooms                   23699 non-null int64
ceiling_height          14504 non-null float64
floors_total            23613 non-null float64
living_area             21796 non-null float64
floor                   23699 non-null int64
is_apartment            2775 non-null object
studio                  23699 non-null bool
open_plan               23699 non-null bool
kitchen_area            21421 non-null float64
balcony                 12180 non-null float64
locality_name           23650 non-null object
airports_nearest        18157 non-null float64
cityCenters_nearest     18180 non-null float64
parks_around3000        18181 non-null float64
parks_nearest           8079 non-null float64
ponds_around3000        18181 non-null float64
ponds_nearest           9110 non-null float64
days_exposition         20518 non-null float64
dtypes: bool(2), float64(14), int64(3), object(3)
memory usage: 3.7+ MB
```

Видим, что в датасете присутствуют данные нескольких типов: вещественные (`float64`), булевые (`bool`) и строковые (`object`). Также видно, что в датасете есть явные пропуски.

```
for col in df.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = df[df[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
total_images - 0
last_price - 0
total_area - 0
first_day_exposition - 0
rooms - 0
ceiling_height - 9195
floors_total - 86
living_area - 1903
floor - 0
is_apartment - 20924
studio - 0
open_plan - 0
kitchen_area - 2278
balcony - 11519
locality_name - 49
airports_nearest - 5542
cityCenters_nearest - 5519
parks_around3000 - 5518
parks_nearest - 15620
ponds_around3000 - 5518
```

```
ponds_nearest - 14589
days_exposition - 3181

# Основные статистические характеристки набора данных
df.describe()
```

```
       total_images    last_price    total_area         rooms
ceiling_height  \
count  23699.000000  2.369900e+04  23699.000000  23699.000000
14504.000000
mean       9.858475  6.541549e+06     60.348651      2.070636
2.771499
std        5.682529  1.088701e+07     35.654083      1.078405
1.261056
min        0.000000  1.219000e+04     12.000000      0.000000
1.000000
25%        6.000000  3.400000e+06     40.000000      1.000000
2.520000
50%        9.000000  4.650000e+06     52.000000      2.000000
2.650000
75%       14.000000  6.800000e+06     69.900000      3.000000
2.800000
max       50.000000  7.630000e+08    900.000000     19.000000
100.000000

       floors_total   living_area         floor  kitchen_area
balcony  \
count  23613.000000  21796.000000  23699.000000  21421.000000
12180.000000
mean      10.673824     34.457852      5.892358     10.569807
1.150082
std        6.597173     22.030445      4.885249      5.905438
1.071300
min        1.000000      2.000000      1.000000      1.300000
0.000000
25%        5.000000     18.600000      2.000000      7.000000
0.000000
50%        9.000000     30.000000      4.000000      9.100000
1.000000
75%       16.000000     42.300000      8.000000     12.000000
2.000000
max       60.000000    409.700000     33.000000    112.000000
5.000000

       airports_nearest  cityCenters_nearest  parks_around3000
parks_nearest  \
count      18157.000000         18180.000000      18181.000000
8079.000000
mean       28793.672193         14191.277833          0.611408
490.804555
```

```
std        12630.880622          8608.386210          0.802074
342.317995
min            0.000000           181.000000          0.000000
1.000000
25%        18585.000000          9238.000000          0.000000
288.000000
50%        26726.000000         13098.500000          0.000000
455.000000
75%        37273.000000         16293.000000          1.000000
612.000000
max        84869.000000         65968.000000          3.000000
3190.000000

       ponds_around3000  ponds_nearest  days_exposition
count      18181.000000    9110.000000     20518.000000
mean           0.770255     517.980900       180.888634
std            0.938346     277.720643       219.727988
min            0.000000      13.000000         1.000000
25%            0.000000     294.000000        45.000000
50%            1.000000     502.000000        95.000000
75%            1.000000     729.000000       232.000000
max            3.000000    1344.000000      1580.000000
```

Очистка данных (к оглавлению)

Пропущенные данные (к оглавлению)

Выведем список параметров датасета и для каждого из них найдём процент null значений.

```
proportion_nans = []
for i in df.columns:
    print('{} : {:.2%}\n'.format(i, df[i].isna().sum()/df.shape[0]))
    proportion_nans.append([i,df[i].isna().sum()/df.shape[0] * 100])
```

```
total_images : 0.00%

last_price : 0.00%

total_area : 0.00%

first_day_exposition : 0.00%

rooms : 0.00%
```

```
ceiling_height : 38.80%

floors_total : 0.36%

living_area : 8.03%

floor : 0.00%

is_apartment : 88.29%

studio : 0.00%

open_plan : 0.00%

kitchen_area : 9.61%

balcony : 48.61%

locality_name : 0.21%

airports_nearest : 23.38%

cityCenters_nearest : 23.29%

parks_around3000 : 23.28%

parks_nearest : 65.91%

ponds_around3000 : 23.28%

ponds_nearest : 61.56%

days_exposition : 13.42%
```

Столбцы, у которых пропущено **>40%** данных, удалим, т.к. в данном датасете они не имеют большого веса, однако иногда такие удаления могут навредить, например, может вырасти количество дубликатов, порой в разы.

Столбцы, у которых пропущено **от 0.1% до 40%** данных, заполним средним по квантилям

```
columns_to_del = []
columns_tofill_mean = []
columns_tofill_median = []
for i in proportion_nans:
    if(i[1] > 40):
        columns_to_del.append(i[0])
```

```
        print('{:19} ({:5.2f}%) => columns_to_del'.format(i[0],i[1]))
    elif (i[1] > 0):
        columns_tofill_median.append(i[0])
        print('{:19} ({:5.2f}%) =>
columns_tofill_median'.format(i[0],i[1]))
```

```
ceiling_height      (38.80%) => columns_tofill_median
floors_total        ( 0.36%) => columns_tofill_median
living_area         ( 8.03%) => columns_tofill_median
is_apartment        (88.29%) => columns_to_del
kitchen_area        ( 9.61%) => columns_tofill_median
balcony             (48.61%) => columns_to_del
locality_name       ( 0.21%) => columns_tofill_median
airports_nearest    (23.38%) => columns_tofill_median
cityCenters_nearest (23.29%) => columns_tofill_median
parks_around3000    (23.28%) => columns_tofill_median
parks_nearest       (65.91%) => columns_to_del
ponds_around3000    (23.28%) => columns_tofill_median
ponds_nearest       (61.56%) => columns_to_del
days_exposition     (13.42%) => columns_tofill_median
```

Проблемы конкретных столбцов (к оглавлению)

Проверим данные на наличие дубликатов. Для начала проверим параметр `locality_name`.

```
print(f"Уникальных значений параметра 'locality_name':
{df['locality_name'].unique().size}.")
print(f"Количество записей в датасете: {df.shape[0]}.")
```

```
Уникальных значений параметра 'locality_name': 365.
Количество записей в датасете: 23699.
```

Видна проблема дублирования данных. Например посёлок Бугры и поселок Бугры это одно и то же, но в таблице отмечены как разные значения. Необходимо лемматизировать данные в этом столбце и убрать тип населенного пункта в значениях.

Обработка пропущенных значений (к оглавлению)

```
for i in columns_tofill_median:
    print("{:19} - {}".format(i, df[i].isna().sum()))
```

```
ceiling_height      - 9195
floors_total        - 86
living_area         - 1903
kitchen_area        - 2278
locality_name       - 49
```

```
airports_nearest    - 5542
cityCenters_nearest - 5519
parks_around3000    - 5518
ponds_around3000    - 5518
days_exposition     - 3181
```

Столбец **locality_name** имеет строковый тип данных и заполнить его медианой не получится

Так как данные пропущены всего лишь в 49 строках можно просто их удалить

```
df.dropna(subset=['locality_name'], inplace=True)
df.reset_index(inplace=True, drop=True)
df['locality_name'].isna().sum()
```

```
0
```

Обработаем созданный ранее список **columns_tofill_median**, пропуски в котором заполним значениями квантилей и медианой(1/3 заполняется первым квантилем, 1/3 заполняется третим квартилем, 1/3 заполняется медианой). Заменять просто на медиану некоторые параметры - это очень сильно усреднять показатели. Если высота потолков спокойно выдержит такую процедуру, т.к. это не настолько уникальный параметр, то жилую площадь, площадь кухни лучше заменять в зависимости от какого-нибудь коэффициента, или в зависимости от категорий.

Можно выдвинуть гипотезу, почему такие пропуски образовались. Например, days_exposition пропуск может свидетельствовать, что или объявление еще не снято, или было снято в течении суток после выставления.

```
for i in columns_tofill_median:
    if(df[i].dtype != object):
        Q1 = df[i].quantile(0.25)
        Q3 = df[i].quantile(0.75)
        med = df[i].median()
        c = int(df[i].isna().sum() * 0.33)
        df[i].fillna(Q1,limit = c ,inplace=True)
        df[i].fillna(Q3,limit = c ,inplace=True)
        df[i].fillna(med,limit = c ,inplace=True)
        df.dropna(subset=[i], inplace=True)

df.reset_index(inplace=True)
```

```
for i in columns_tofill_median:
    print("{:19} - {}".format(i, df[i].isna().sum()))
```

```
ceiling_height      - 0
floors_total        - 0
living_area         - 0
```

```
kitchen_area        - 0
locality_name       - 0
airports_nearest    - 0
cityCenters_nearest - 0
parks_around3000    - 0
ponds_around3000    - 0
days_exposition     - 0
```

Видим, что пропущенные данные теперь отсутствуют. Обработаем созданный ранее список **columns_to_del**, столбцы входящие в этот список надо удалить.

```
print(columns_to_del)
```

```
['is_apartment', 'balcony', 'parks_nearest', 'ponds_nearest']
```

```
df = df.drop(columns=columns_to_del)
df.columns
```

```
Index(['index', 'total_images', 'last_price', 'total_area',
       'first_day_exposition', 'rooms', 'ceiling_height',
'floors_total',
       'living_area', 'floor', 'studio', 'open_plan', 'kitchen_area',
       'locality_name', 'airports_nearest', 'cityCenters_nearest',
       'parks_around3000', 'ponds_around3000', 'days_exposition'],
      dtype='object')
```

Изменение типов данных столбцов (к оглавлению)

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23252 entries, 0 to 23251
Data columns (total 19 columns):
index                   23252 non-null int64
total_images            23252 non-null int64
last_price              23252 non-null float64
total_area              23252 non-null float64
first_day_exposition    23252 non-null object
rooms                   23252 non-null int64
ceiling_height          23252 non-null float64
floors_total            23252 non-null float64
living_area             23252 non-null float64
floor                   23252 non-null int64
studio                  23252 non-null bool
open_plan               23252 non-null bool
kitchen_area            23252 non-null float64
locality_name           23252 non-null object
airports_nearest        23252 non-null float64
cityCenters_nearest     23252 non-null float64
```

```
parks_around3000        23252 non-null float64
ponds_around3000        23252 non-null float64
days_exposition         23252 non-null float64
dtypes: bool(2), float64(11), int64(4), object(2)
memory usage: 3.1+ MB
```

Изменим типы данных таким образом

- total_images ✓
- last_price float64 ➡ int64 (Стоимость в рублях)
- total_area float64 ➡ int64 (Площадь в квадратных метрах)
- first_day_exposition ✓
- rooms ✓
- ceiling_height ✓
- floors_total float64 ➡ int64 (Кол-во этажей целочислено)
- living_area float64 ➡ int64 (Жилая площадь в квадратных метрах)
- floor ✓
- studio ✓
- open_plan ✓
- kitchen_area float64 ➡ int64 (Площадь в квадратных метрах)
- locality_name ✓
- airports_nearest float64 ➡ int64 (Расстояние в метрах)
- cityCenters_nearest float64 ➡ int64 (Расстояние в метрах)
- parks_around3000 float64 ➡ int64 (Кол-во парков целочислено)
- ponds_around3000 float64 ➡ int64 (Кол-во прудов целочислено)
- days_exposition float64 ➡ int64 (Кол-во дней целочислено)

```
df['last_price'] = df['last_price'].astype(int)
df['total_area'] = df['total_area'].astype(int)
df['floors_total'] = df['floors_total'].astype(int)
df['living_area'] = df['living_area'].astype(int)
df['kitchen_area'] = df['kitchen_area'].astype(int)
df['airports_nearest'] = df['airports_nearest'].astype(int)
df['cityCenters_nearest'] = df['cityCenters_nearest'].astype(int)
df['parks_around3000'] = df['parks_around3000'].astype(int)
df['ponds_around3000'] = df['ponds_around3000'].astype(int)
df['days_exposition'] = df['days_exposition'].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23252 entries, 0 to 23251
Data columns (total 19 columns):
index                   23252 non-null int64
total_images            23252 non-null int64
last_price              23252 non-null int32
total_area              23252 non-null int32
first_day_exposition    23252 non-null object
```

```
rooms                   23252 non-null int64
ceiling_height          23252 non-null float64
floors_total            23252 non-null int32
living_area             23252 non-null int32
floor                   23252 non-null int64
studio                  23252 non-null bool
open_plan               23252 non-null bool
kitchen_area            23252 non-null int32
locality_name           23252 non-null object
airports_nearest        23252 non-null int32
cityCenters_nearest     23252 non-null int32
parks_around3000        23252 non-null int32
ponds_around3000        23252 non-null int32
days_exposition         23252 non-null int32
dtypes: bool(2), float64(1), int32(10), int64(4), object(2)
memory usage: 2.2+ MB
```

Добавление новых вычисленных столбцов (к оглавлению)

Добавим столбец с ценой квадратного метра.

```
df['price_p_m'] = (df['last_price'] / df['total_area']).astype(int)
df['price_p_m']
```

```
0        120370
1         83750
2         92785
3        408176
4        100000
          ...
23247    106451
23248     91089
23249    101428
23250    131527
23251     74193
Name: price_p_m, Length: 23252, dtype: int32
```

```
df.columns
```

```
Index(['index', 'total_images', 'last_price', 'total_area',
       'first_day_exposition', 'rooms', 'ceiling_height',
'floors_total',
       'living_area', 'floor', 'studio', 'open_plan', 'kitchen_area',
       'locality_name', 'airports_nearest', 'cityCenters_nearest',
       'parks_around3000', 'ponds_around3000', 'days_exposition',
'price_p_m'],
      dtype='object')
```

Добавим столбцы с днем недели, месяцем и годом публикации объявления.

```
df['weekday'] = pd.to_datetime(df['first_day_exposition'], format='%Y-%m-%dT%H:%M:%S').dt.weekday
df['month'] = pd.to_datetime(df['first_day_exposition'], format='%Y-%m-%dT%H:%M:%S').dt.month
df['year'] = pd.to_datetime(df['first_day_exposition'], format='%Y-%m-%dT%H:%M:%S').dt.year

print(df.head(5))
```

```
   index  total_images  last_price  total_area first_day_exposition
rooms  \
0      0            20    13000000         108  2019-03-07T00:00:00
3
1      1             7     3350000          40  2018-12-04T00:00:00
1
2      2            10     5196000          56  2015-08-20T00:00:00
2
3      3             0    64900000         159  2015-07-24T00:00:00
3
4      4             2    10000000         100  2018-06-19T00:00:00
2

   ceiling_height  floors_total  living_area  floor  ...
locality_name  \
0            2.70            16           51      8  ...  Санкт-
Петербург
1            2.52            11           18      1  ...   посёлок
Шушары
2            2.52             5           34      4  ...  Санкт-
Петербург
3            2.52            14           18      9  ...  Санкт-
Петербург
4            3.03            14           32     13  ...  Санкт-
Петербург

   airports_nearest  cityCenters_nearest parks_around3000
ponds_around3000  \
0             18863                16028                1
2
1             12817                18603                0
0
2             21741                13933                1
2
3             28098                 6800                2
3
4             31856                 8098                2
1
```

```
   days_exposition  price_p_m  weekday  month  year
0               45     120370        3      3  2019
1               81      83750        1     12  2018
2              558      92785        3      8  2015
3              424     408176        4      7  2015
4              121     100000        1      6  2018

[5 rows x 23 columns]
```

Добавим столбец с категорией этажа квартиры

```
for i in range(df.shape[0]):
    if(df.loc[i,'floor'] == 1):
        df.loc[i,'category_floor'] = 'Первый'
    elif (df.loc[i,'floor'] == df.loc[i,'floors_total']):
        df.loc[i,'category_floor'] = 'Последний'
    else:
        df.loc[i,'category_floor'] = 'Другой'
print(df['category_floor'])
```

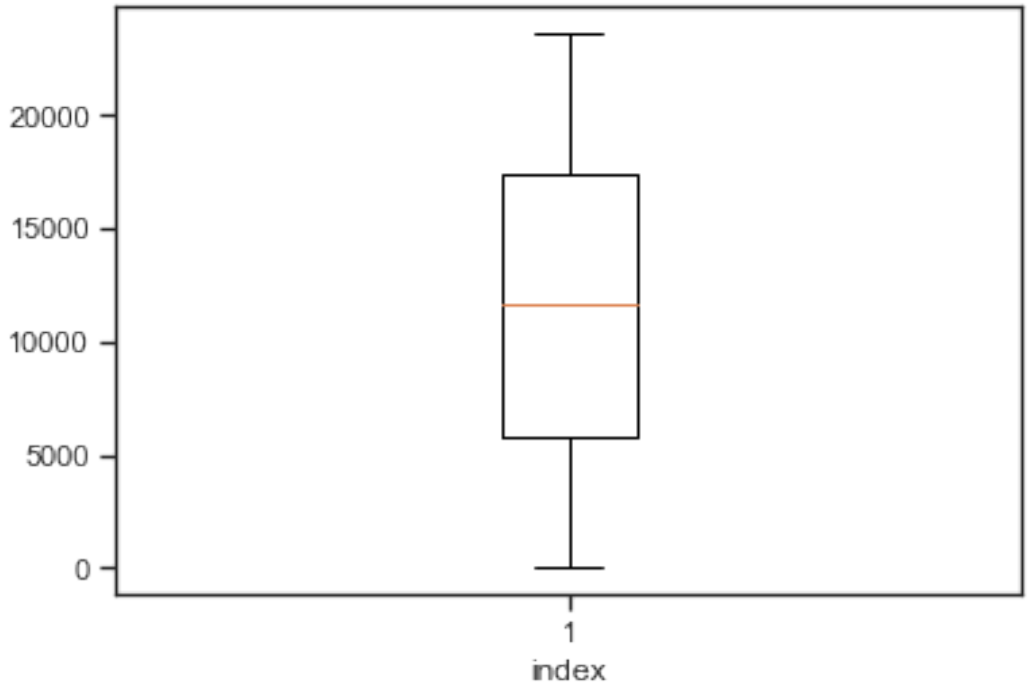
Добавим столбец с соотношением жилой и общей площади, а также отношение площади кухни к общей.

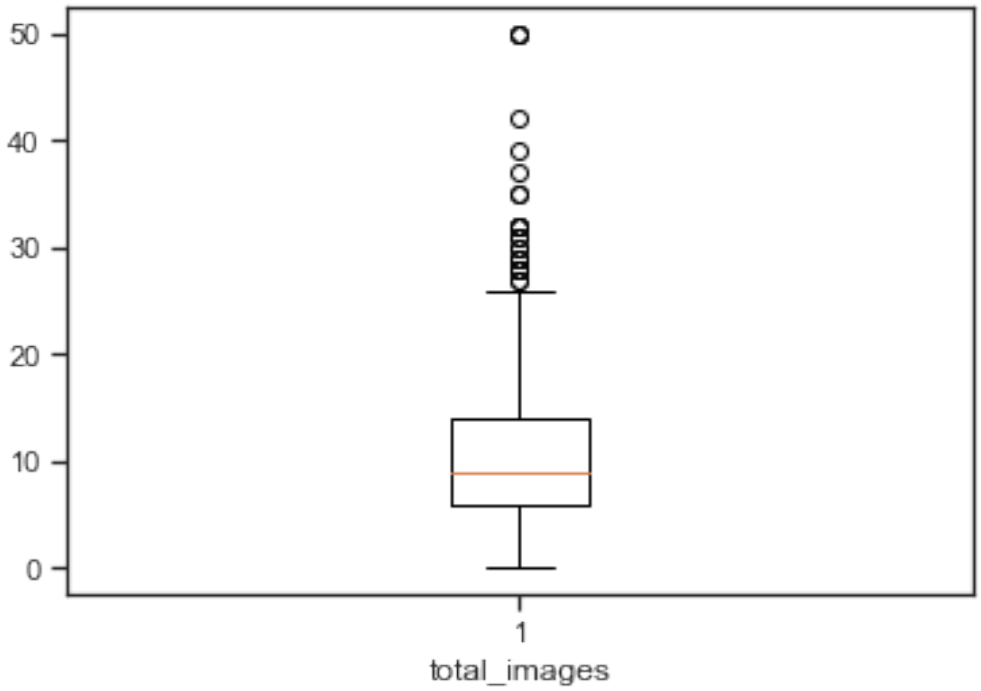
```
df['living_total'] = (df['living_area'] / df['total_area'])
df['kitchen_total'] = (df['kitchen_area'] / df['total_area'])
df[['living_total','kitchen_total']]
```

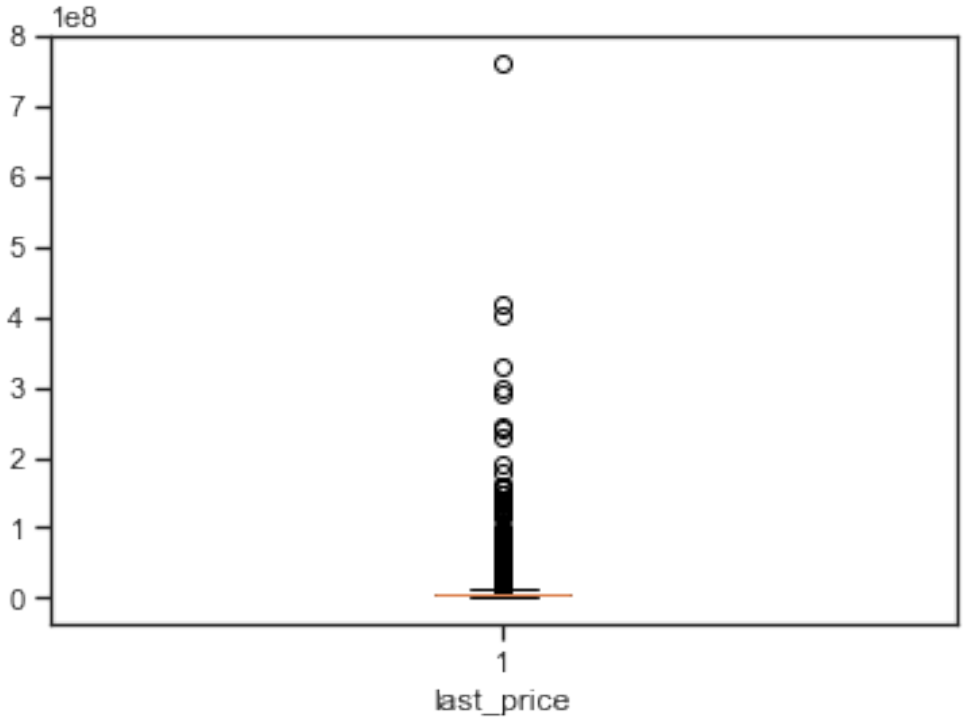
Работа с выбросами в данных (к оглавлению)

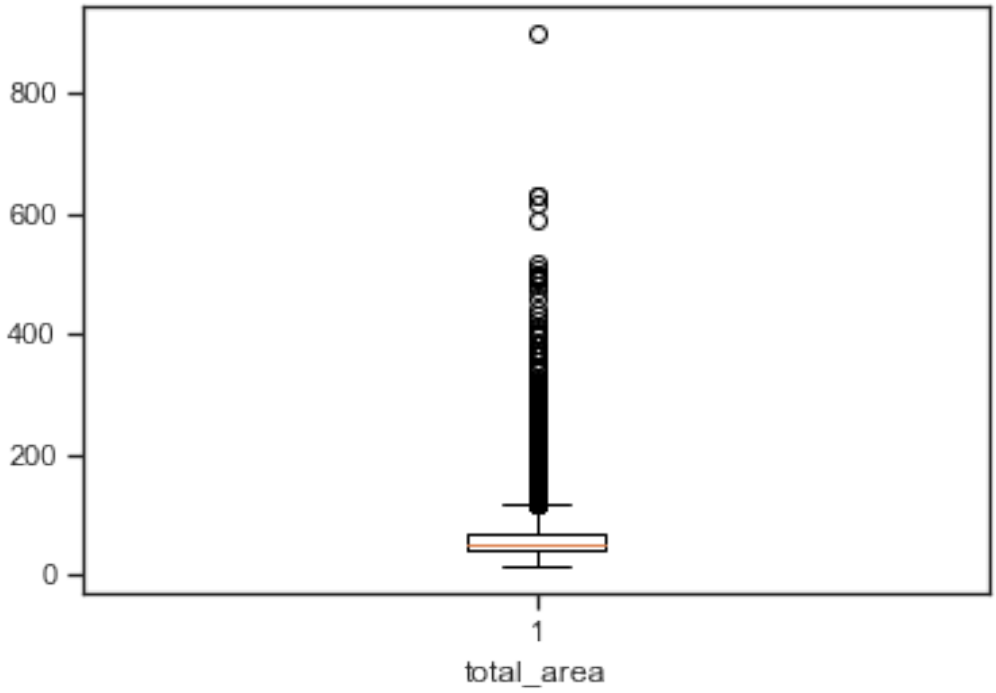
Поработаем с выбросами в данных, уберем редкие и выбивающиеся значения.

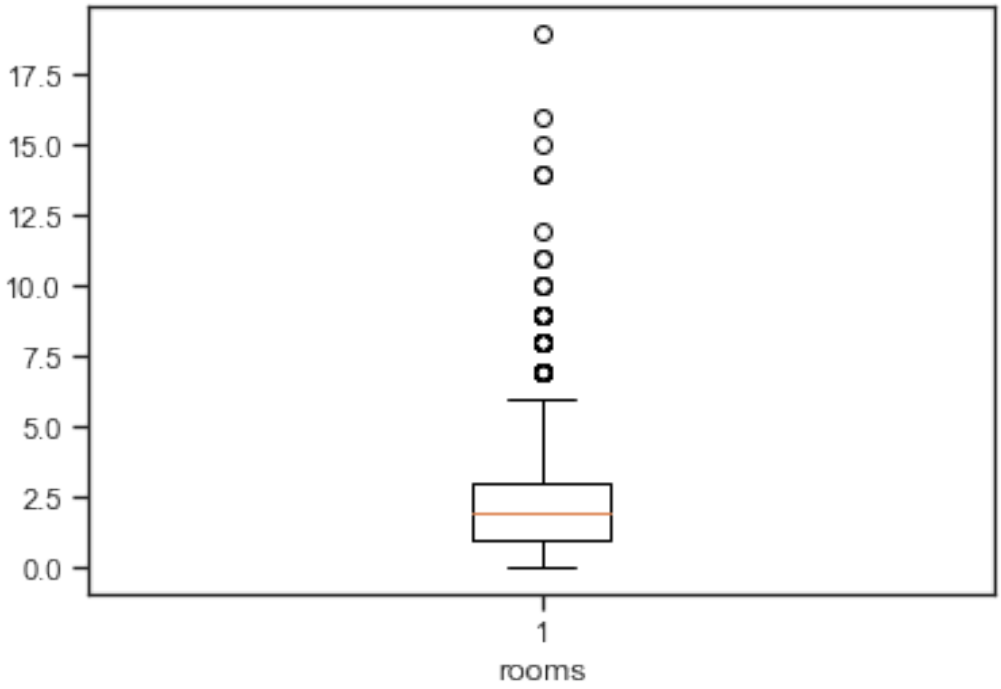
```
new_columns = ['weekday','month','year']
columns_to_del_spike = []
for i in df.columns:
    if(df[i].dtype not in [object, bool]):
        columns_to_del_spike.append(i)
        plt.xlabel(i)
        plt.boxplot(df[i])
        plt.show()
columns_to_del_spike
```

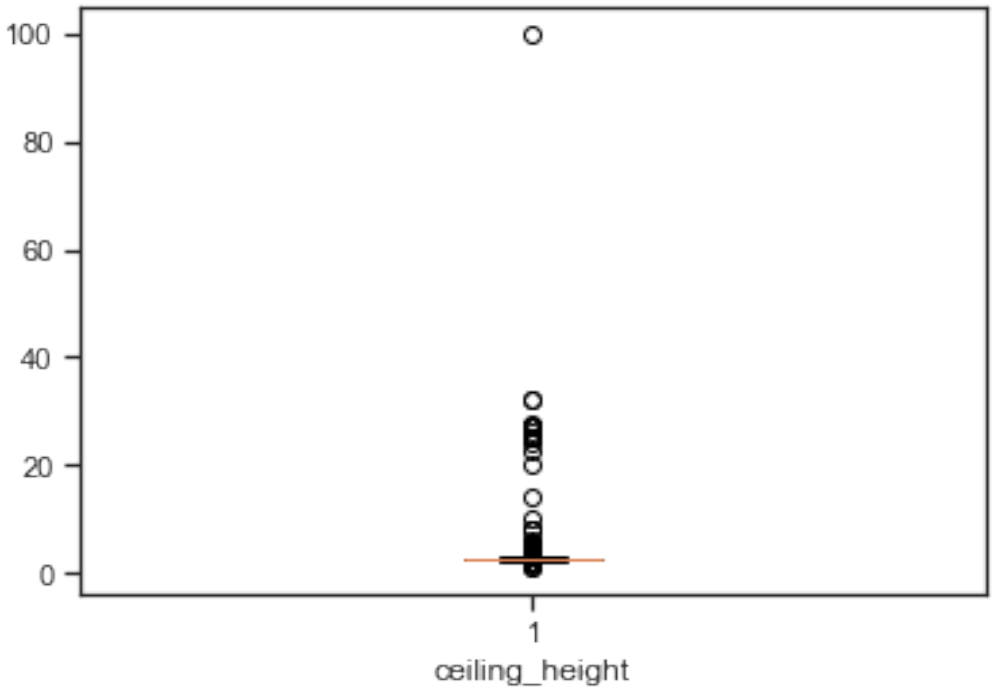

20000
15000
10000
5000
0
1
index

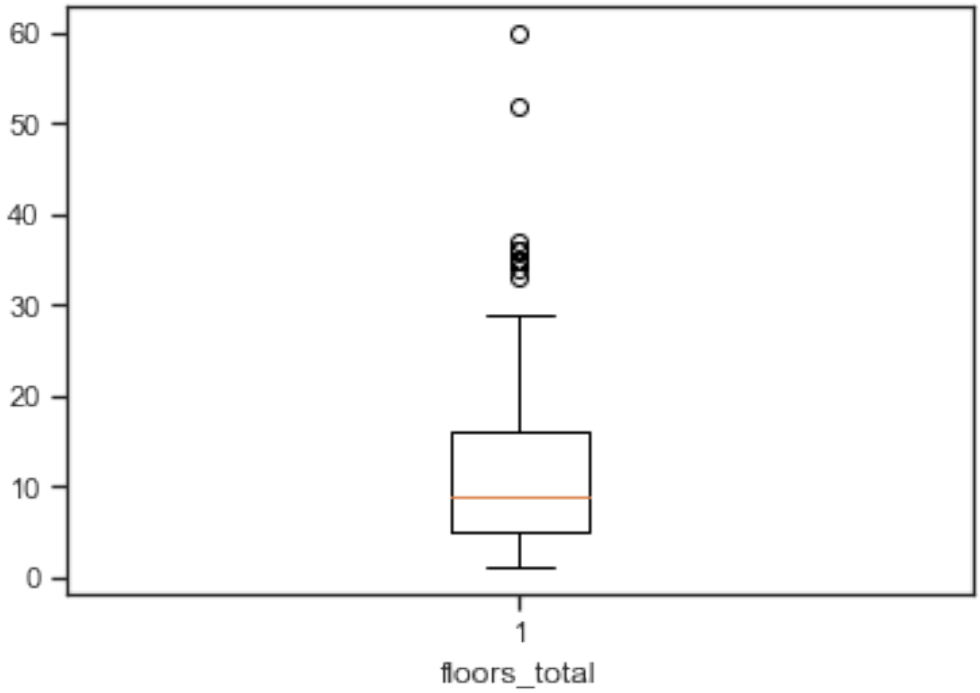

50
40
30
20
10
0
1
total_images

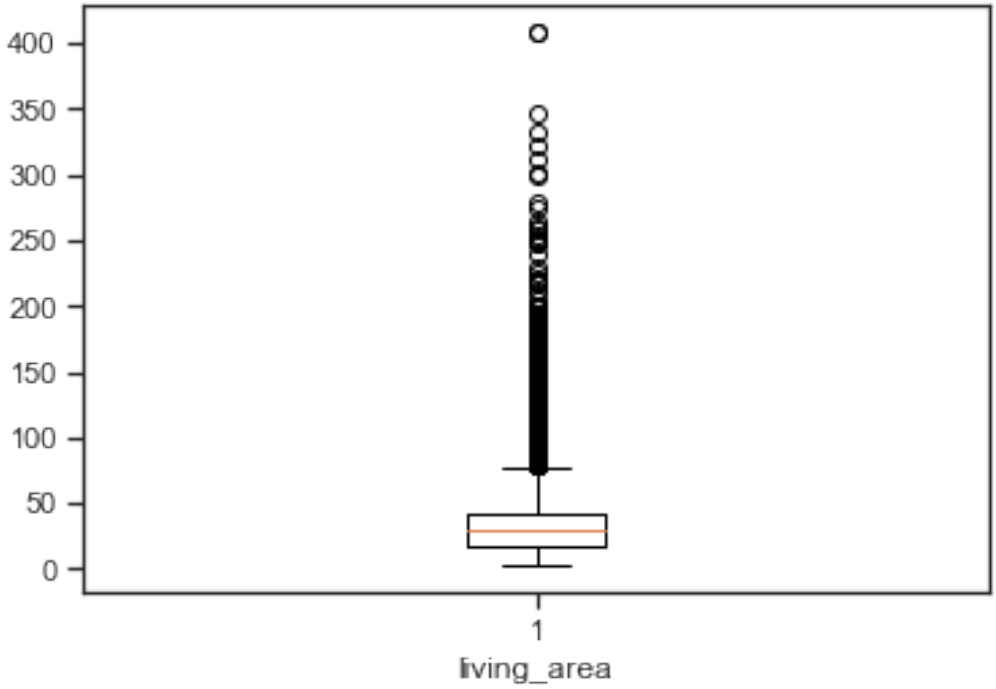

1e8
8
7
6
5
4
3
2
1
0
1
last_price

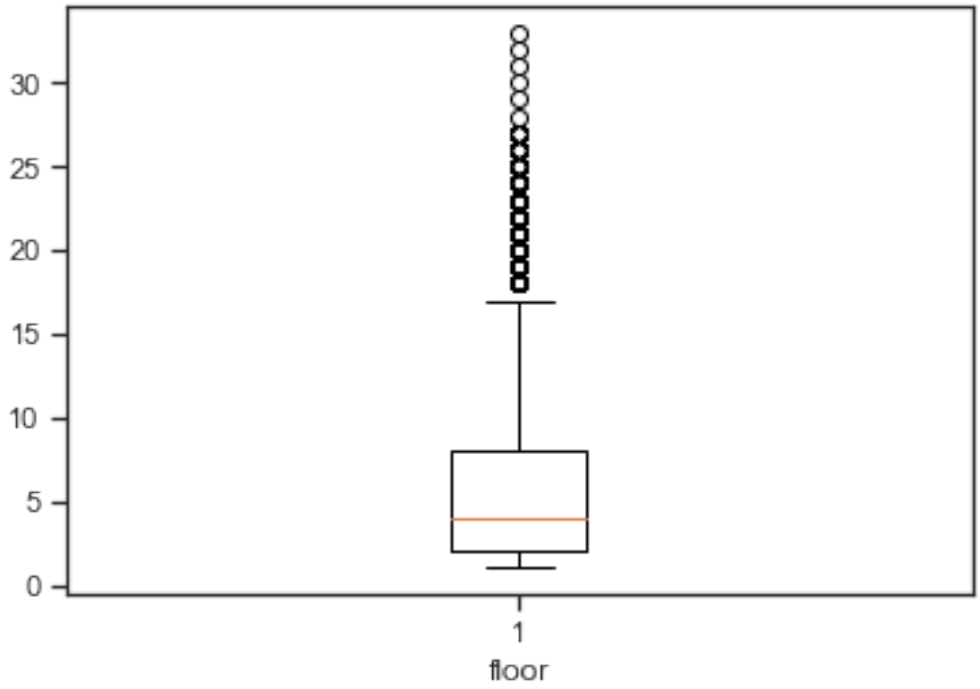

800
600
400
200
0
1
total_area

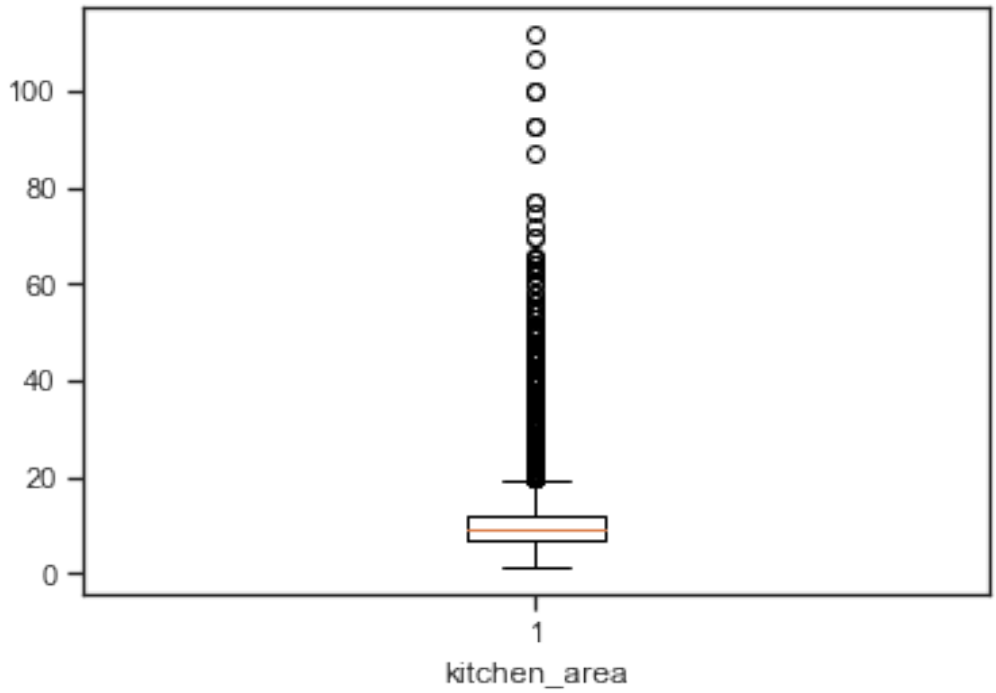

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
rooms

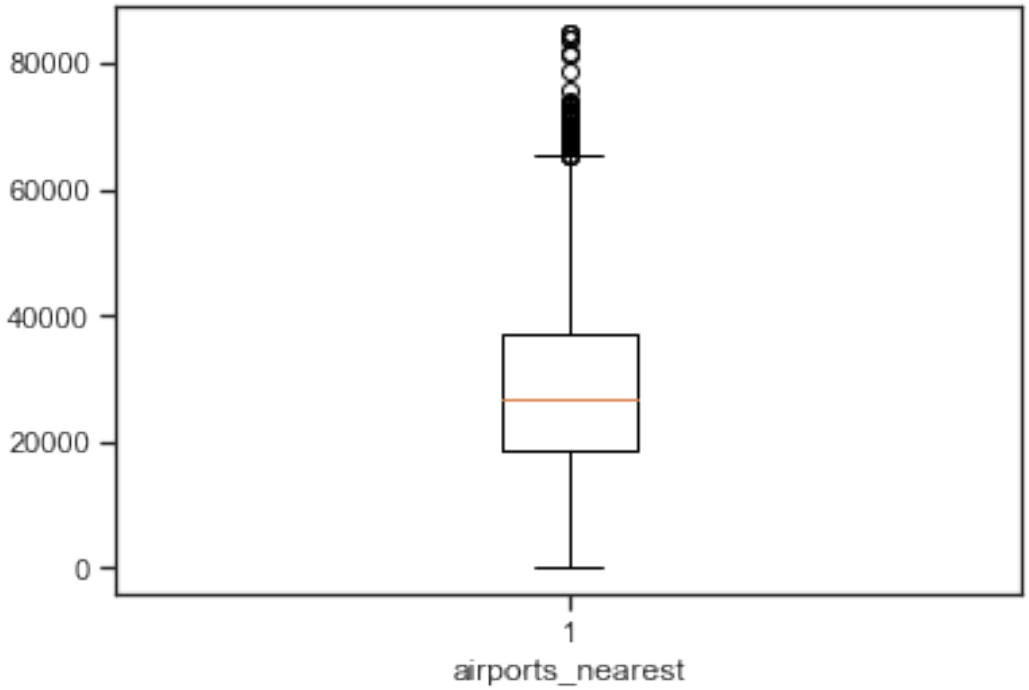

100
80
60
40
20
0
1
ceiling_height

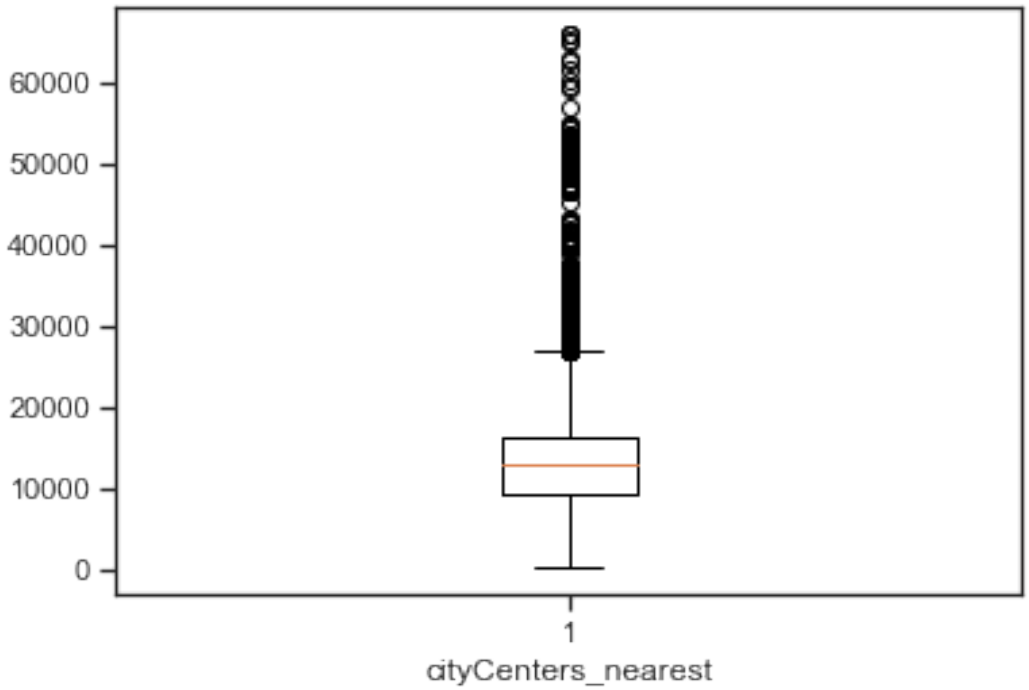

60
50
40
30
20
10
0
1
floors_total

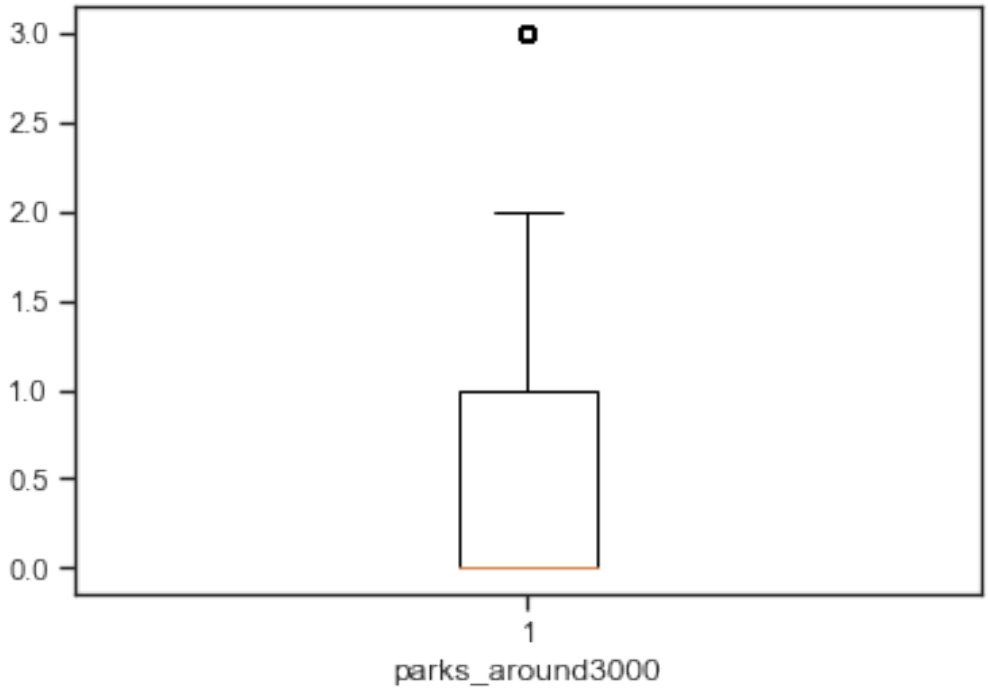

400
350
300
250
200
150
100
50
0
1
living_area

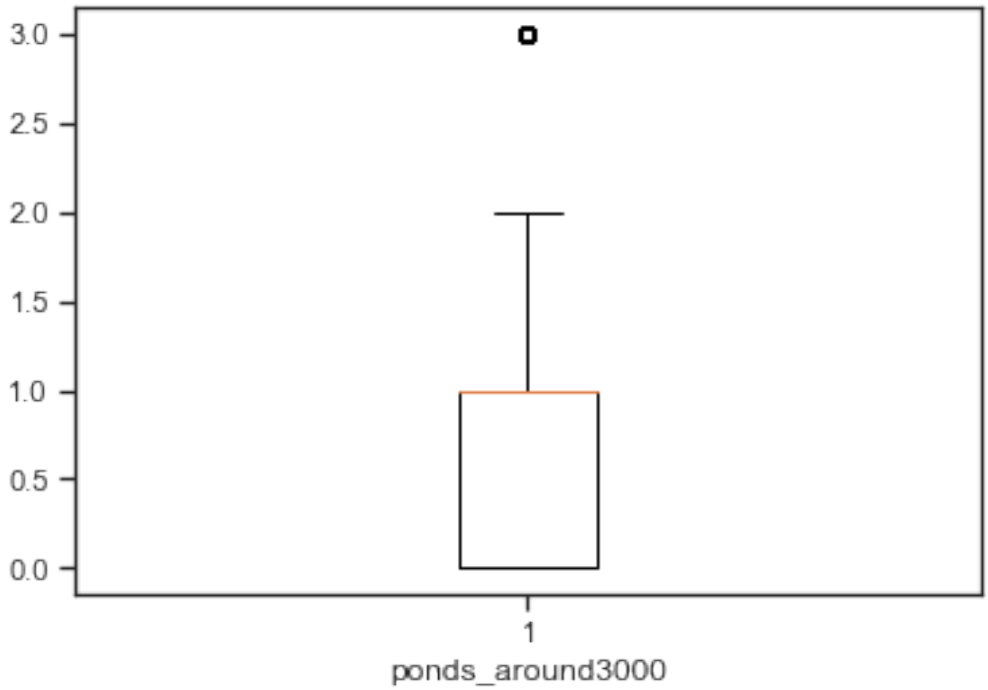

30
25
20
15
10
5
0
1
floor

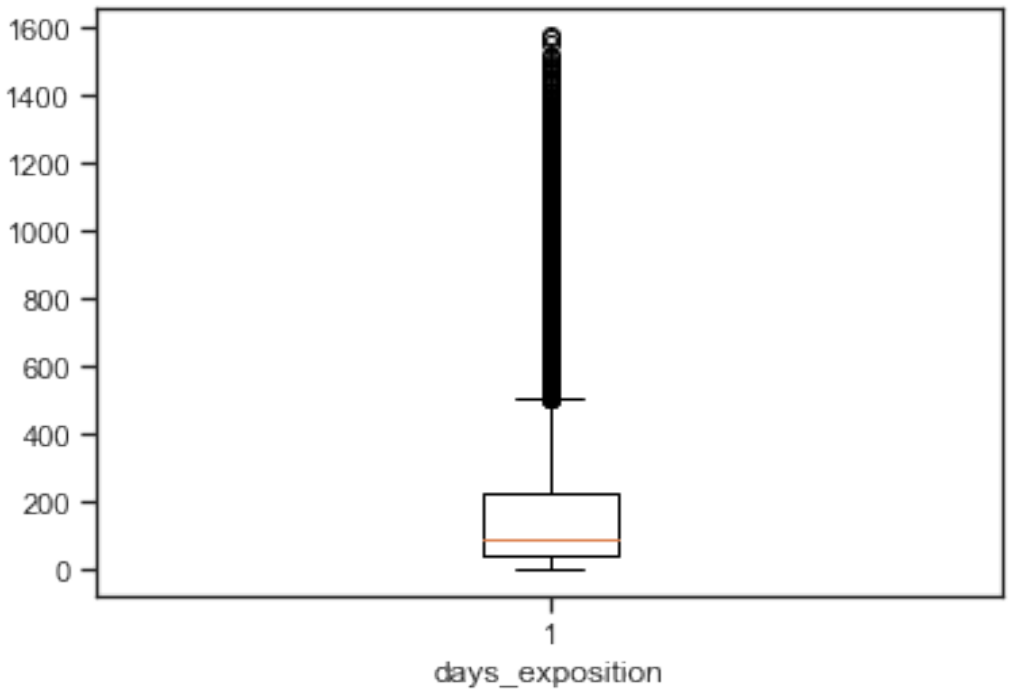

100
80
60
40
20
0
1
kitchen_area

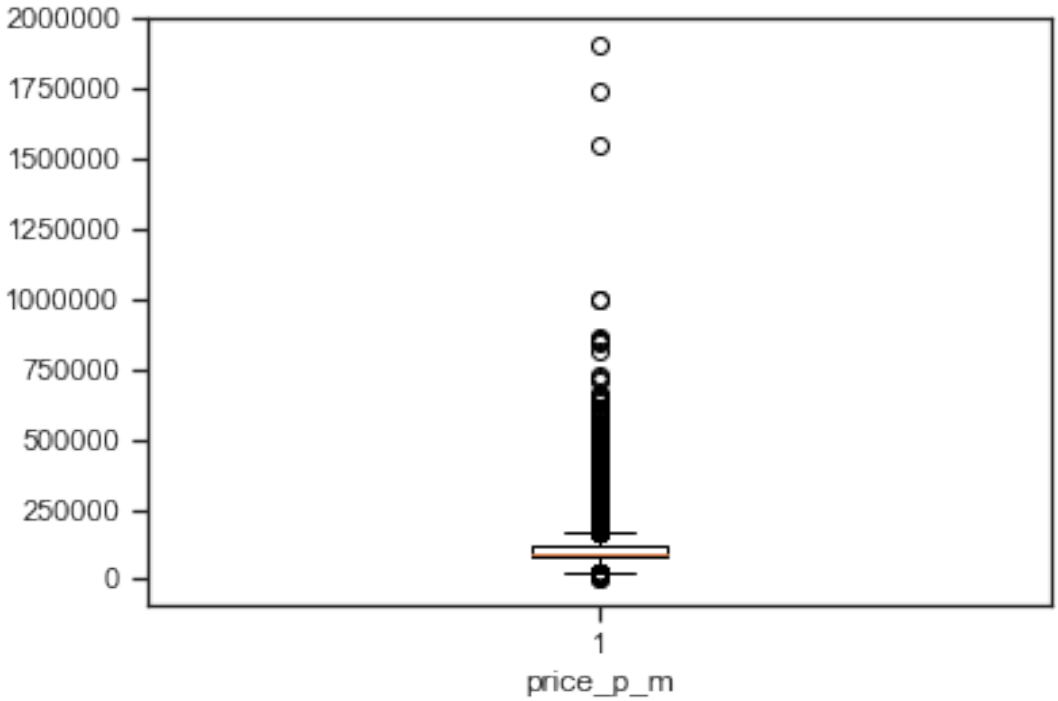

80000
60000
40000
20000
0
1
airports_nearest

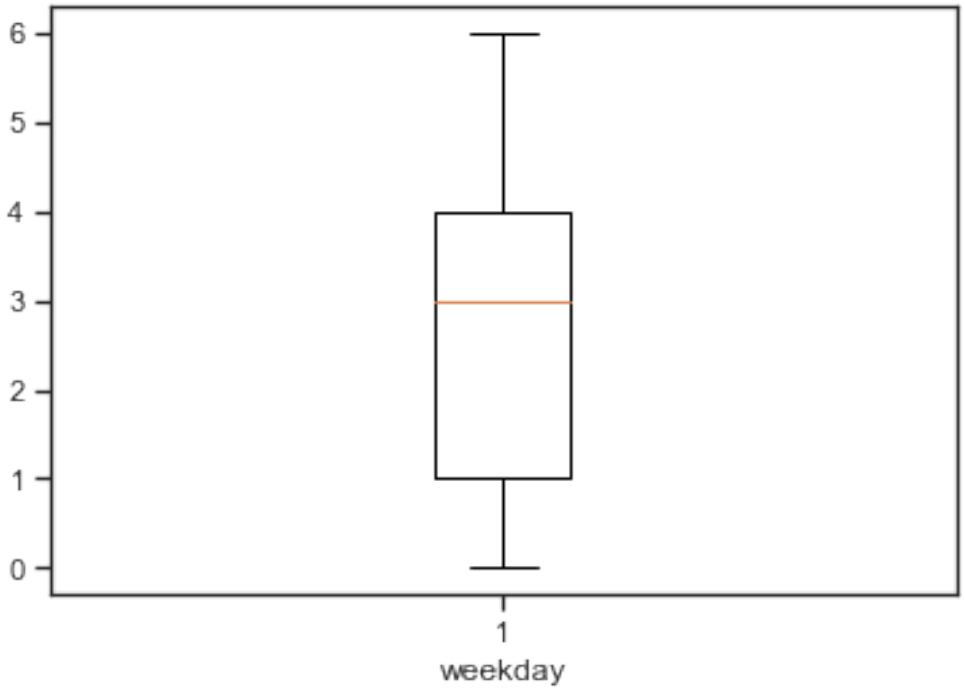

60000
50000
40000
30000
20000
10000
0
1
cityCenters_nearest

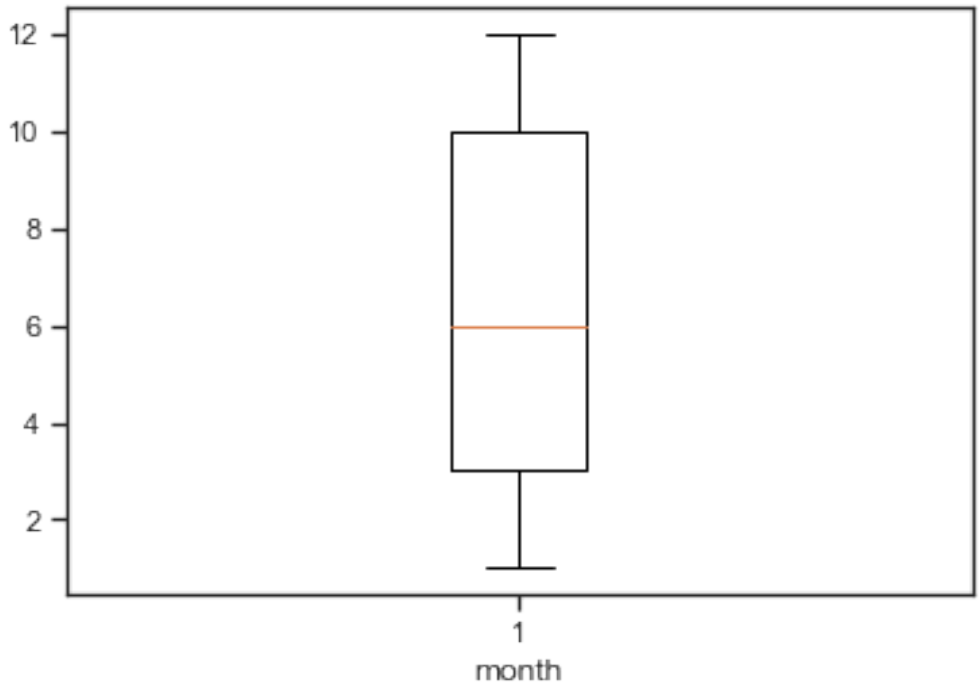

3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
parks_around3000

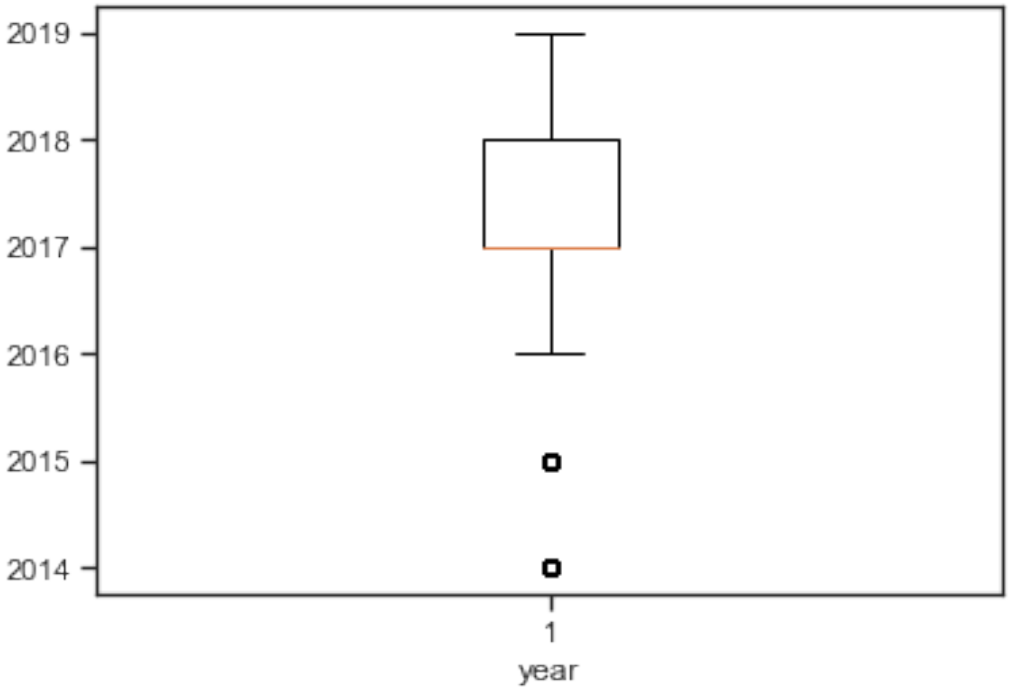

3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
ponds_around3000

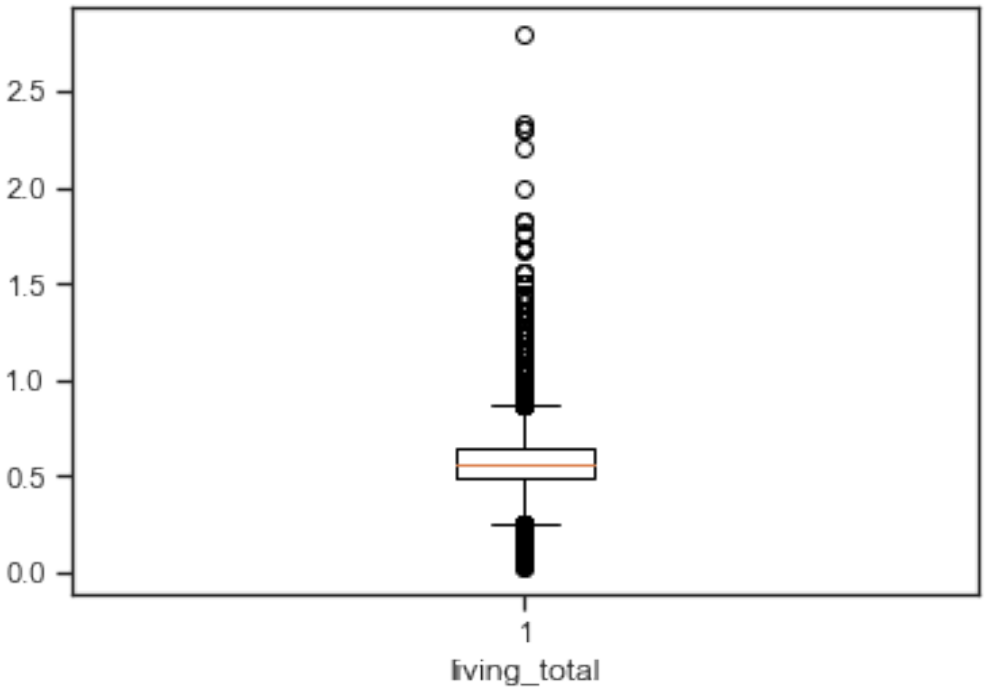

1600
1400
1200
1000
800
600
400
200
0
1
days_exposition

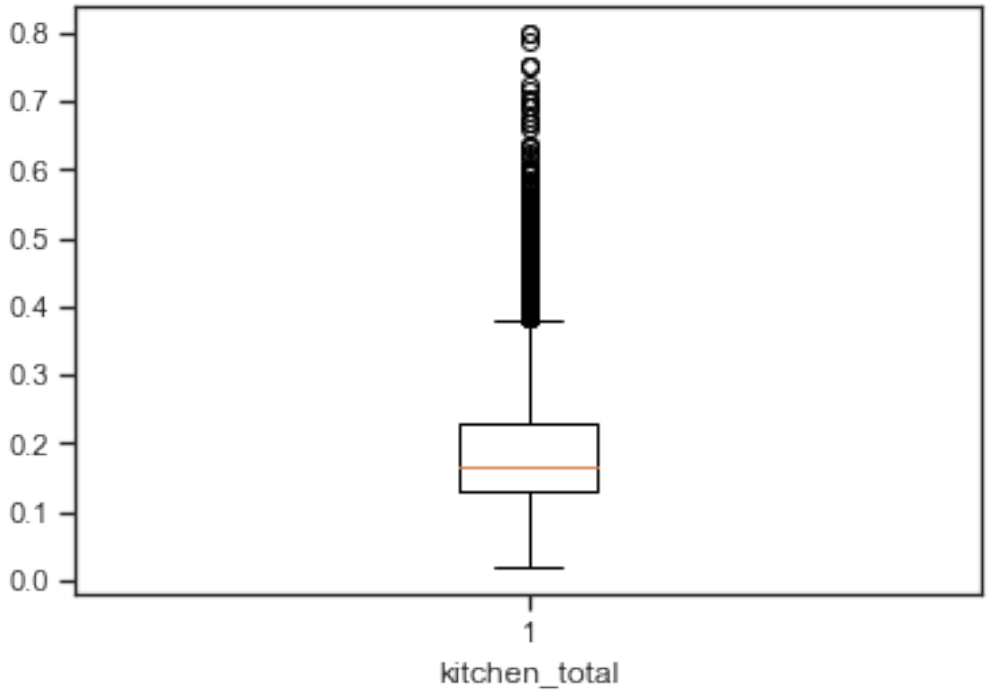

2000000
1750000
1500000
1250000
1000000
750000
500000
250000
0
1
price_p_m


6
5
4
3
2
1
0
1
weekday


12
10
8
6
4
2
1
month


2019
2018
2017
2016
2015
2014
1
year

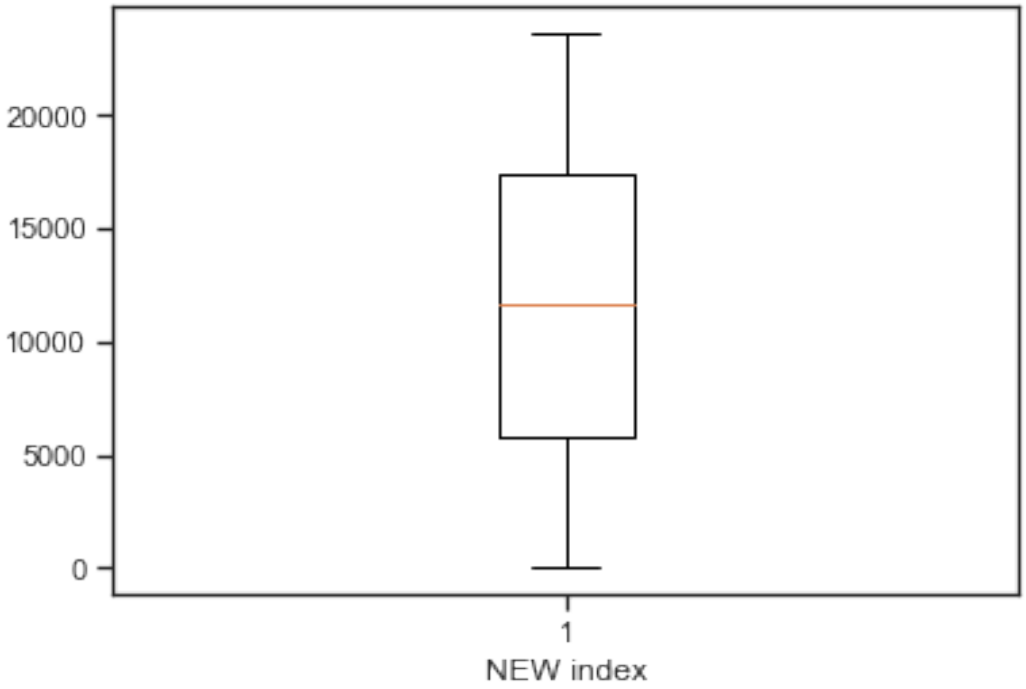

2.5
2.0
1.5
1.0
0.5
0.0
1
living_total

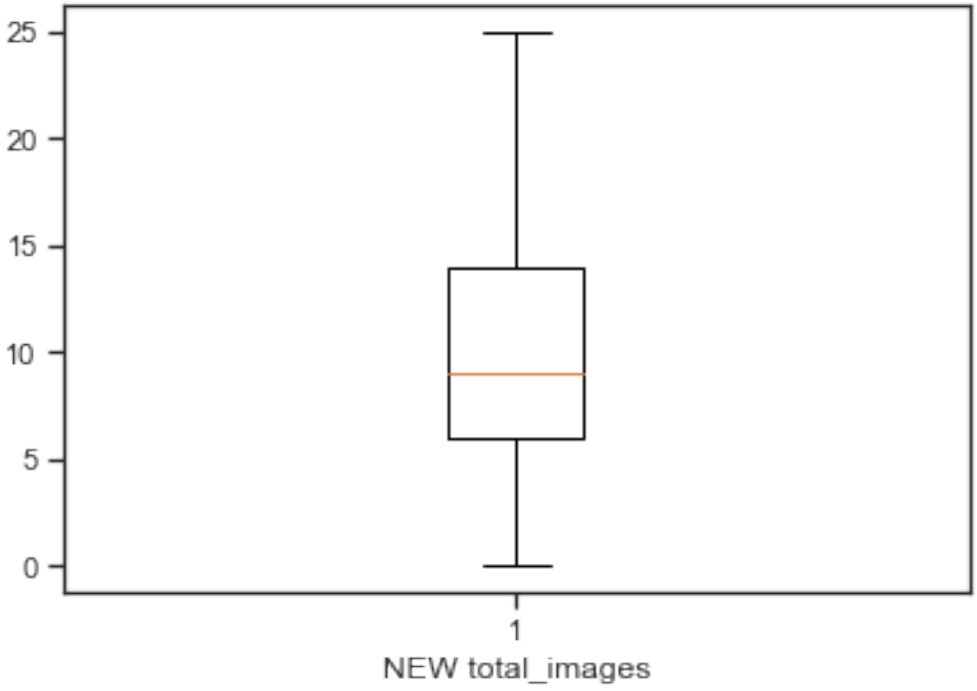


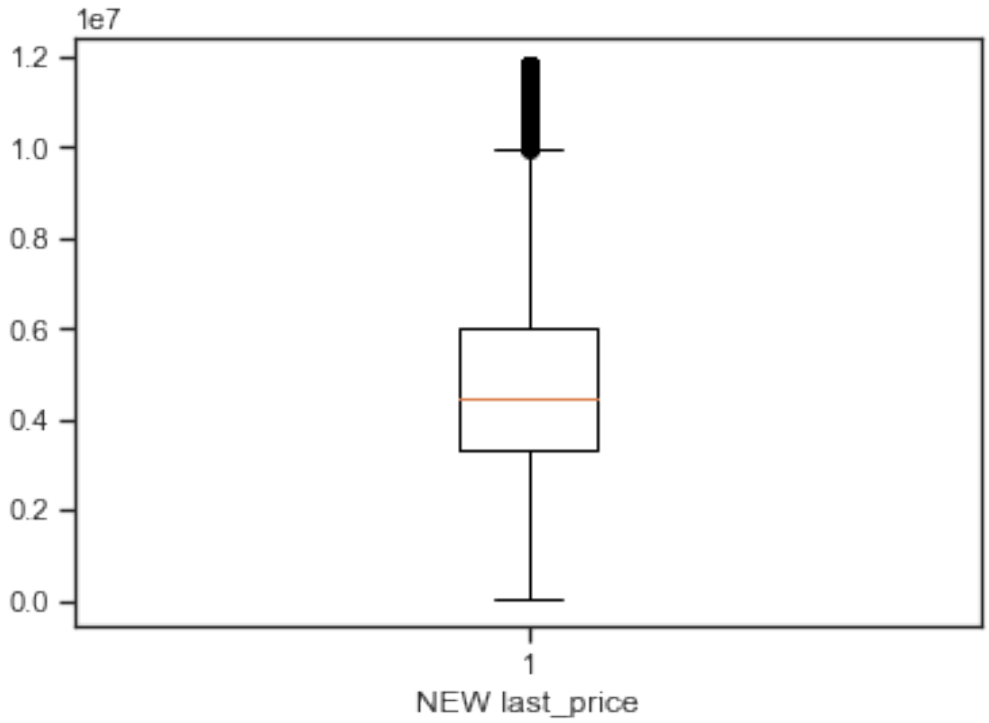

```
['index',
 'total_images',
 'last_price',
 'total_area',
 'rooms',
 'ceiling_height',
 'floors_total',
 'living_area',
 'floor',
 'kitchen_area',
 'airports_nearest',
 'cityCenters_nearest',
 'parks_around3000',
 'ponds_around3000',
 'days_exposition',
 'price_p_m',
 'weekday',
 'month',
 'year',
 'living_total',
 'kitchen_total']
```

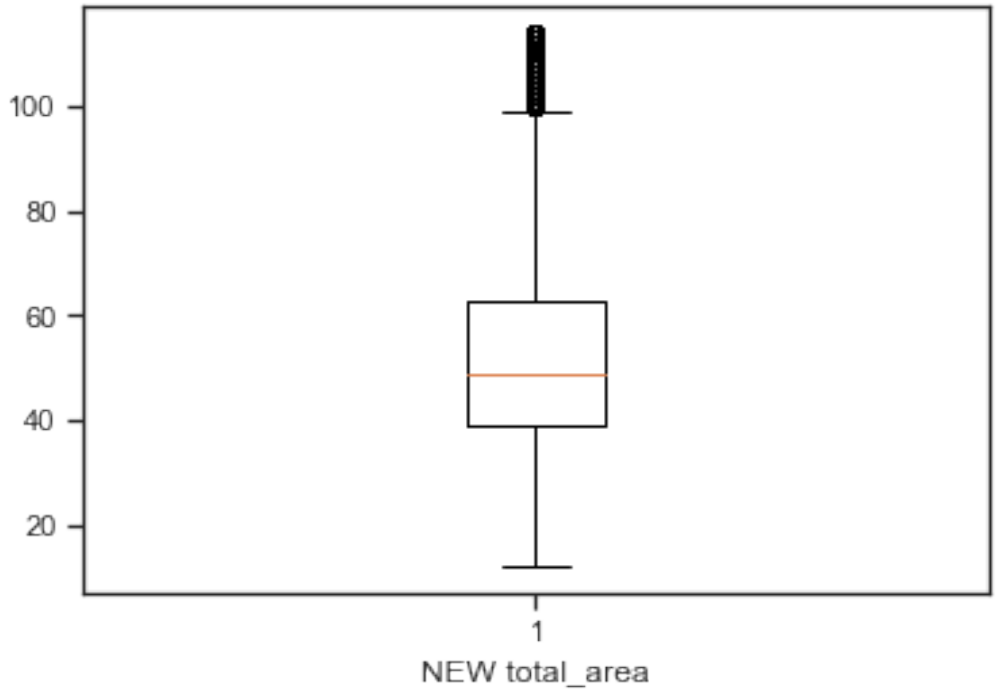
```
df_2 = df
for i in columns_to_del_spike:
    Q1 = df[i].quantile(0.25)
    Q3 = df[i].quantile(0.75)
    IQR = Q3 - Q1
    plt.xlabel("NEW " + i)
    df_2 = df_2[(df[i] > Q1 - 1.5 * IQR) & (df[i] < Q3 + 1.5 * IQR)]
```

```
    plt.boxplot(df_2[i])
    plt.show()
df = df_2
```

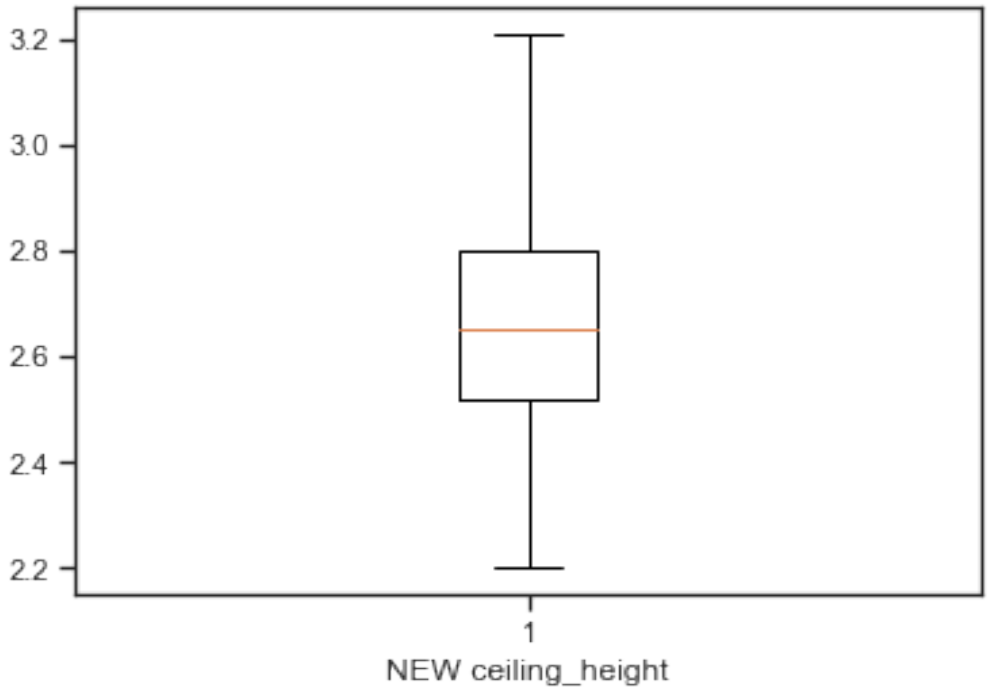


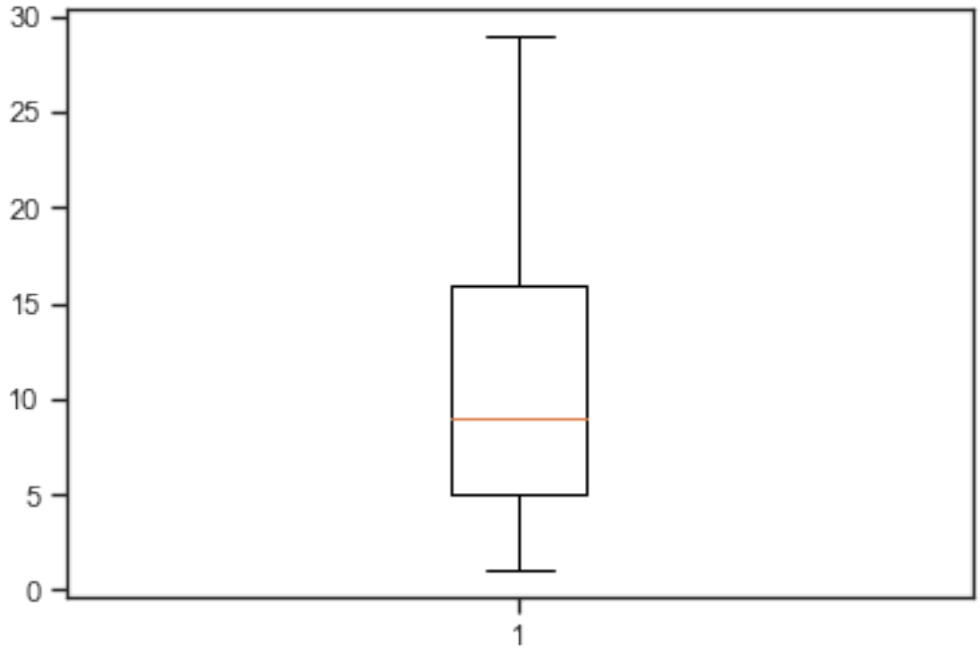


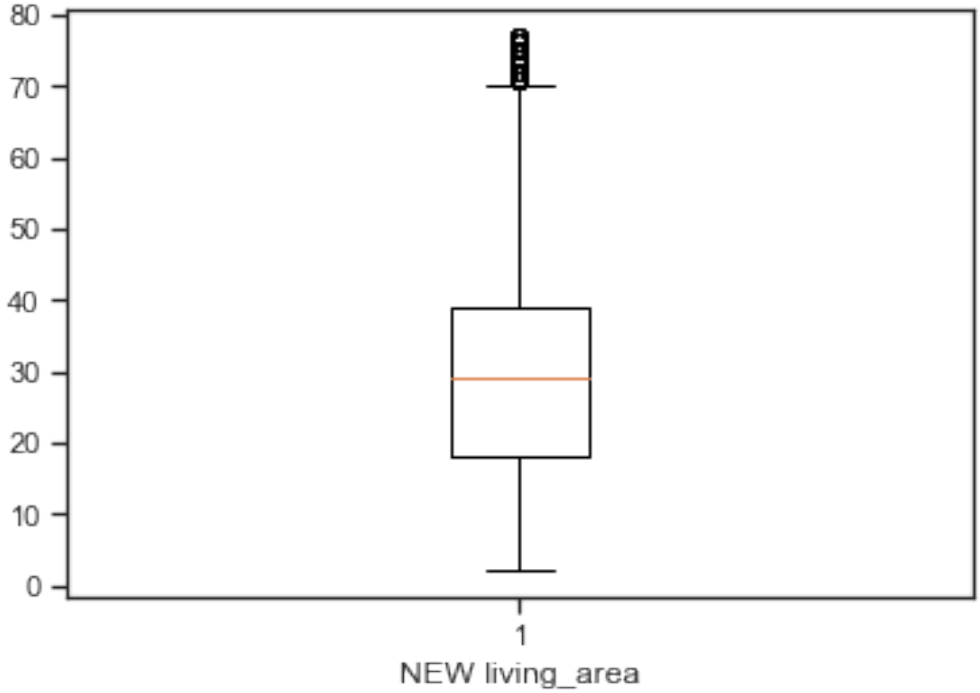

1e7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1
NEW last_price


100
80
60
40
20
1
NEW total_area


5
4
3
2
1
0
1
NEW rooms


3.2
3.0
2.8
2.6
2.4
2.2
1
NEW ceiling_height

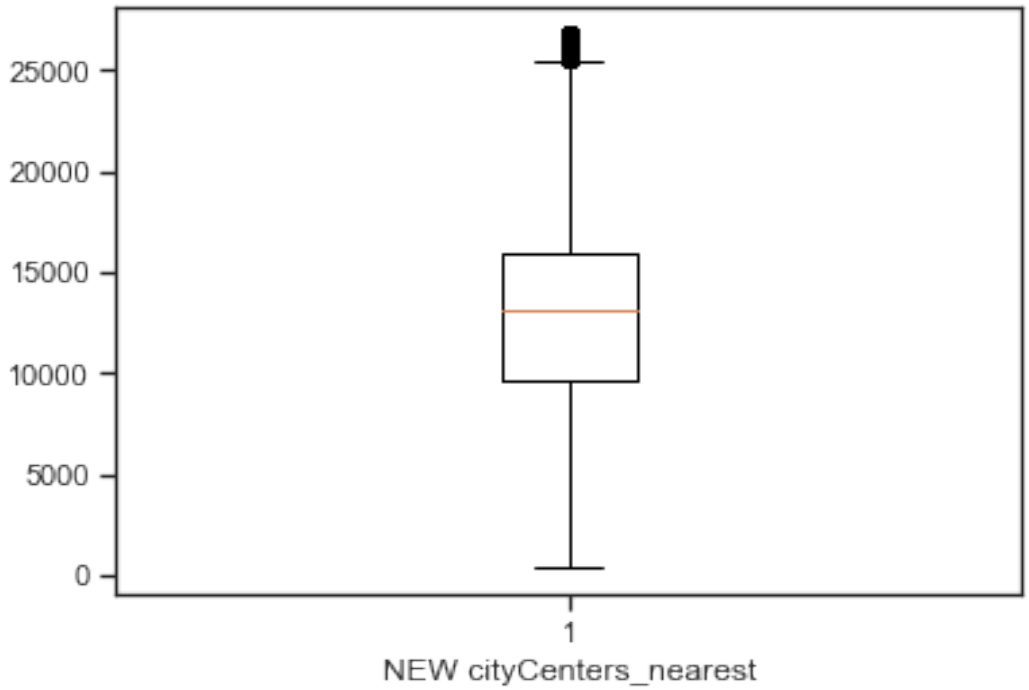

30
25
20
15
10
5
0
1
NEW floors_total

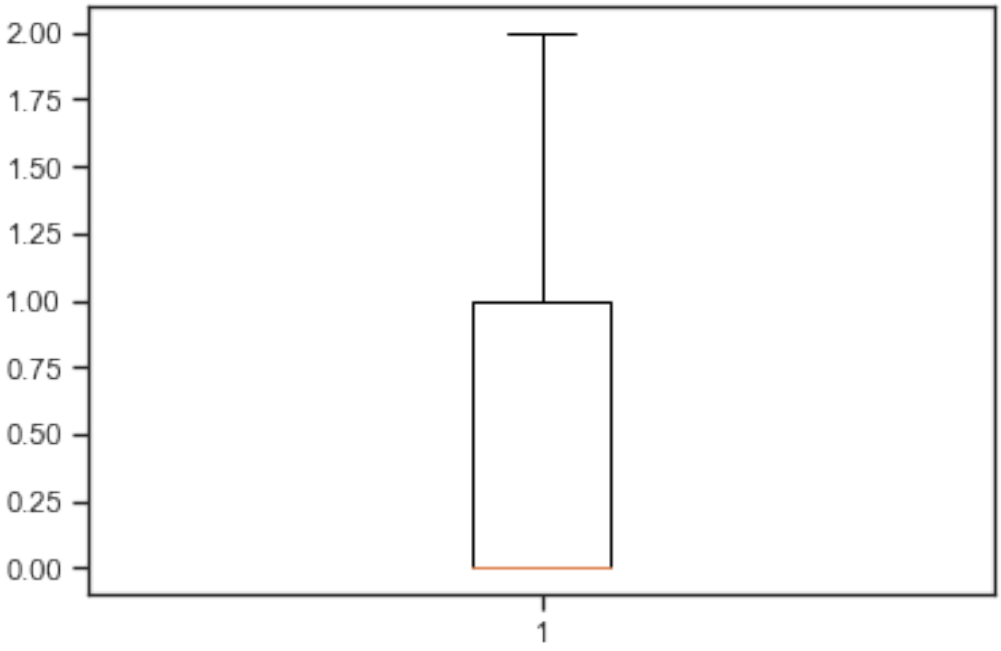

80
70
60
50
40
30
20
10
0
1
NEW living_area

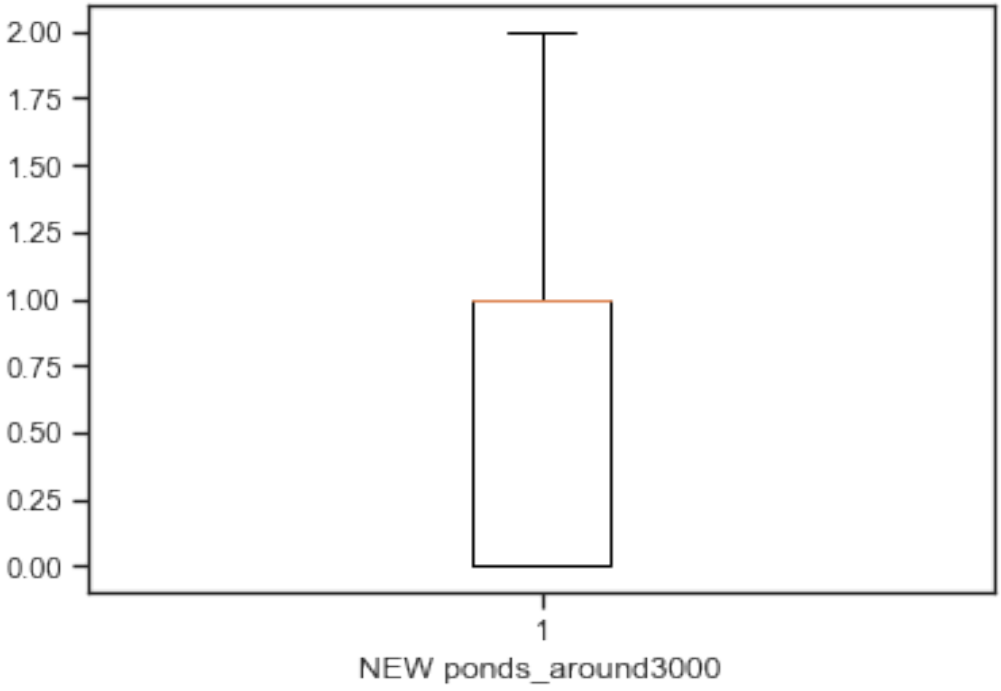

16
14
12
10
8
6
4
2
1
NEW floor

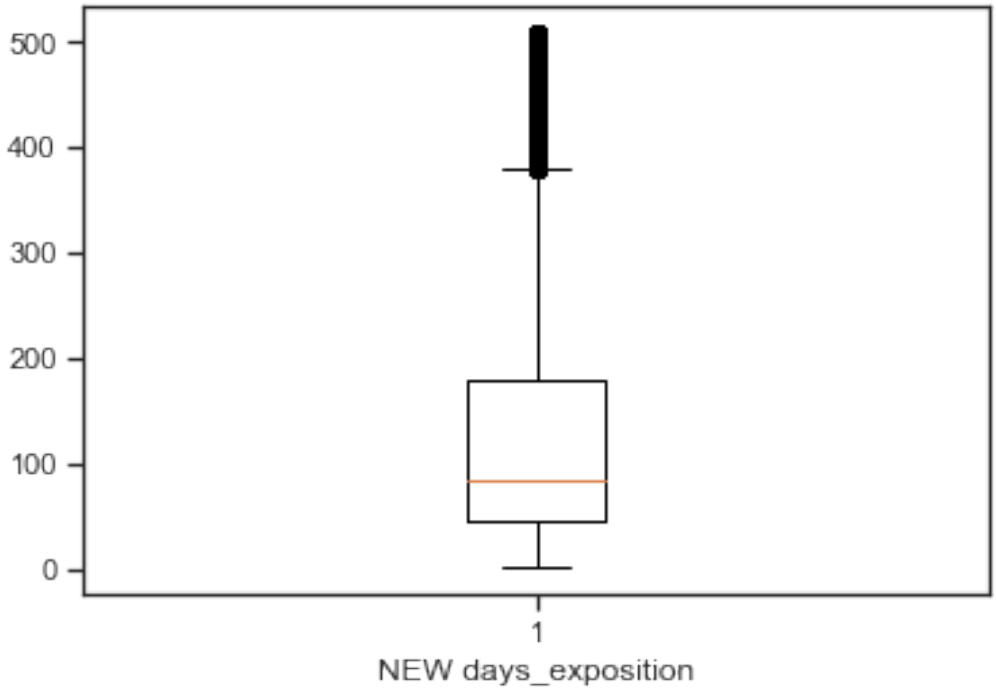

17.5
15.0
12.5
10.0
7.5
5.0
2.5
1
NEW kitchen_area

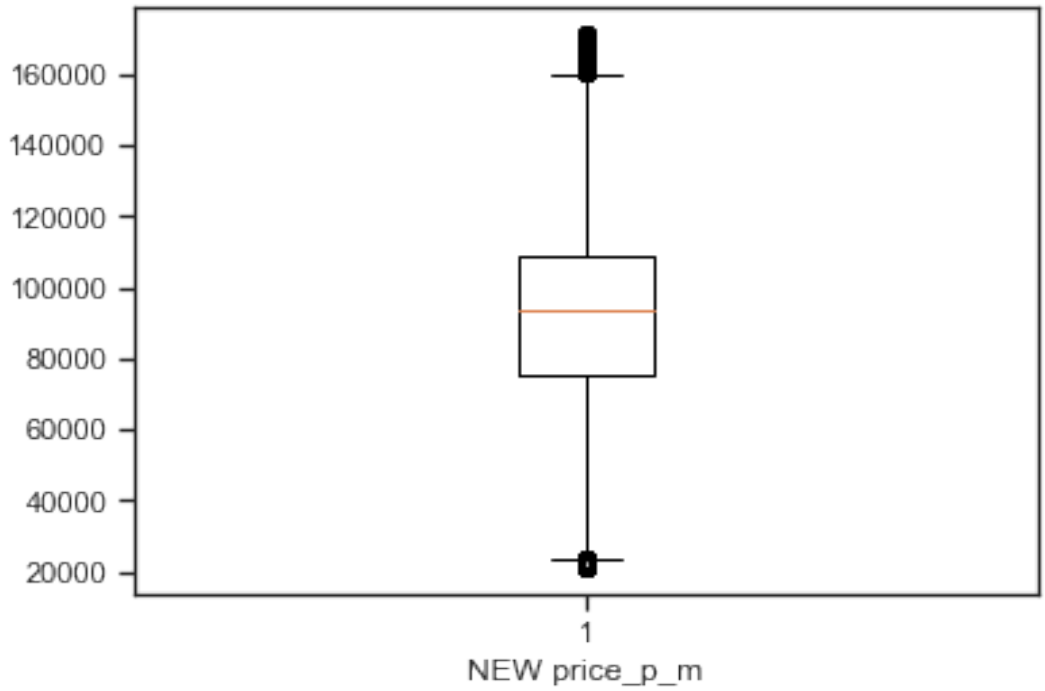

60000
50000
40000
30000
20000
10000
0
1
NEW airports_nearest

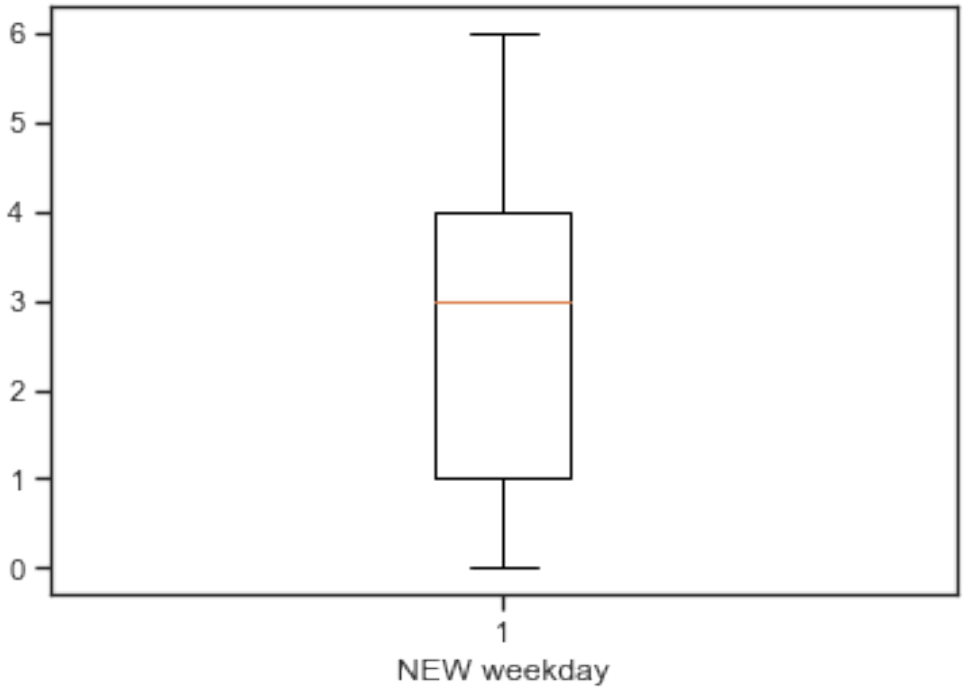

25000
20000
15000
10000
5000
0
1
NEW cityCenters_nearest

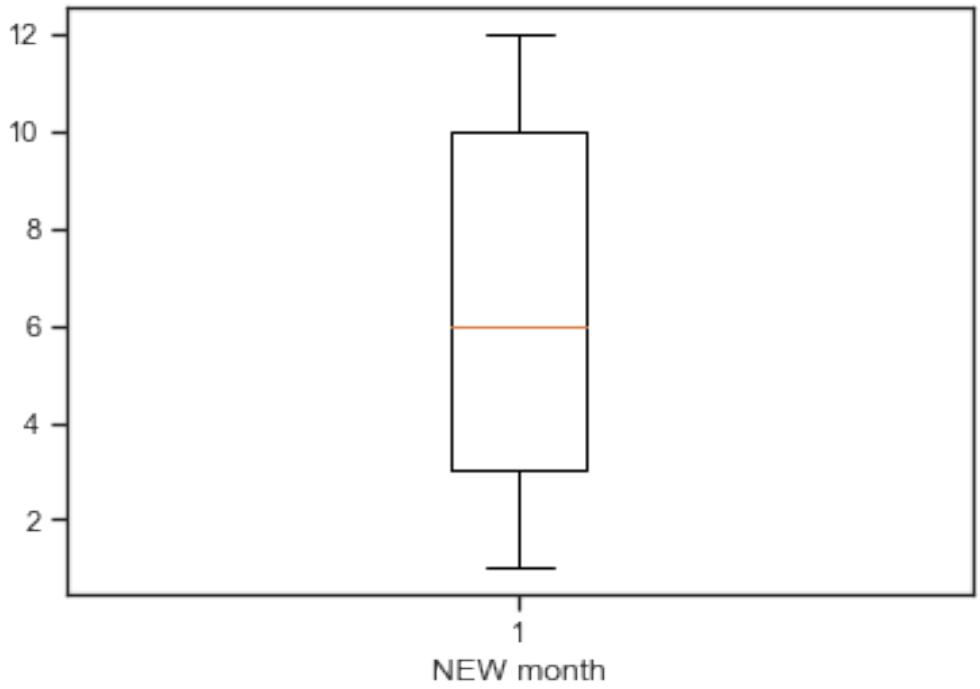

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
1
NEW parks_around3000

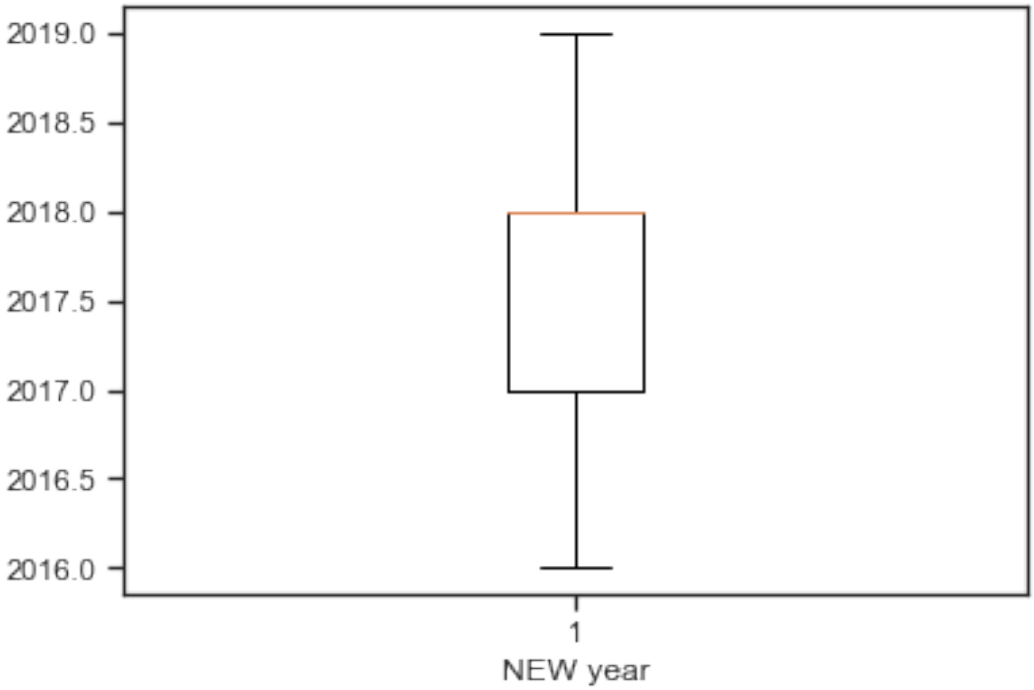

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
1
NEW ponds_around3000

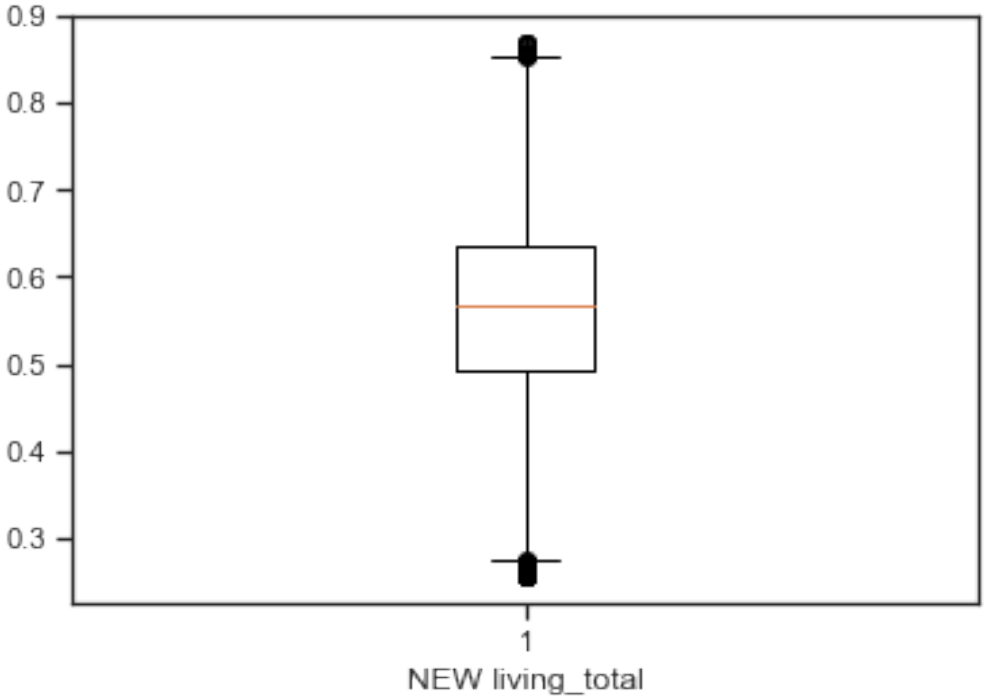

500
400
300
200
100
0
1
NEW days_exposition

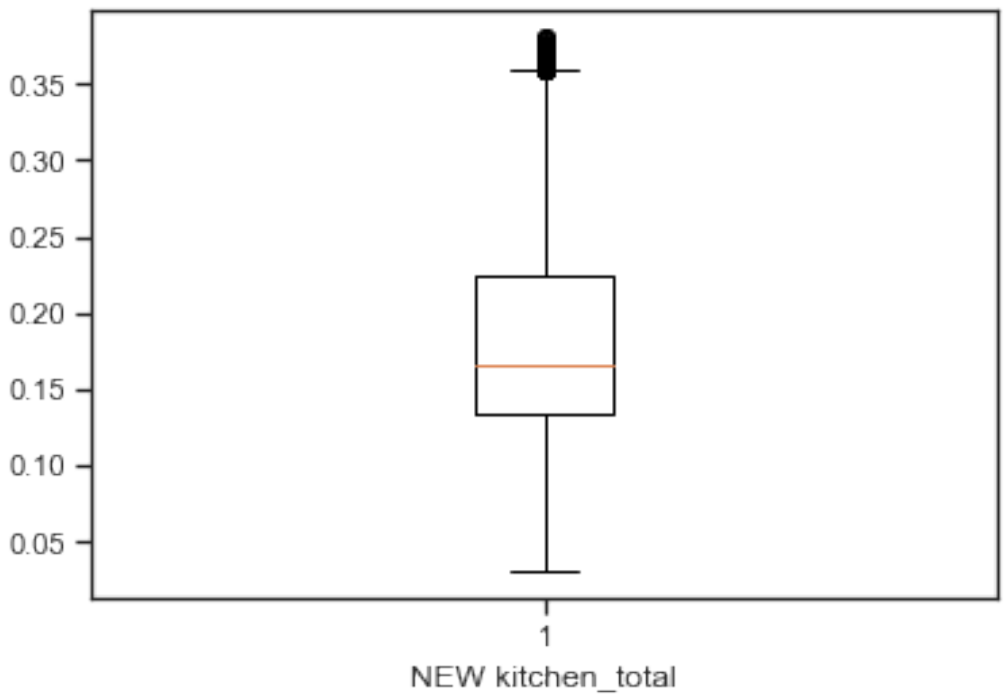

160000
140000
120000
100000
80000
60000
40000
20000
1
NEW price_p_m


6
5
4
3
2
1
0
1
NEW weekday


12
10
8
6
4
2
1
NEW month


2019.0
2018.5
2018.0
2017.5
2017.0
2016.5
2016.0
1
NEW year


0.9
0.8
0.7
0.6
0.5
0.4
0.3
1
NEW living_total

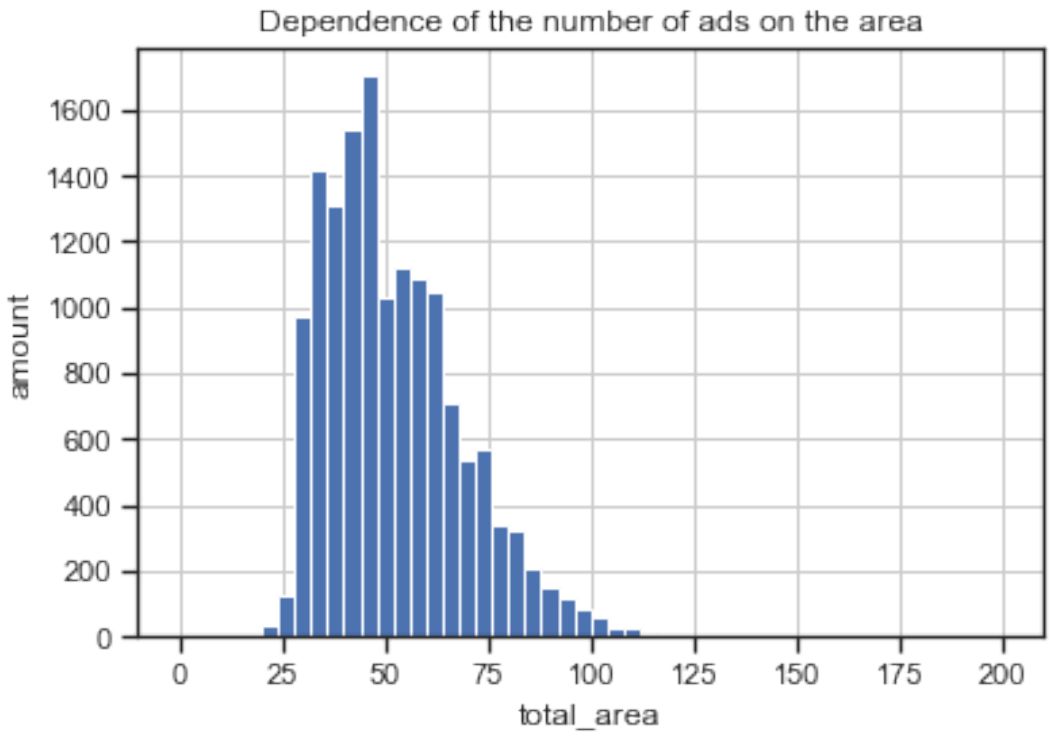


3) Визуальное исследование датасета (к оглавлению)

Исследуем зависимость числа объявлений от различных параметров (к оглавлению)

Визуализируем зависимость числа объявлений от общей площади.

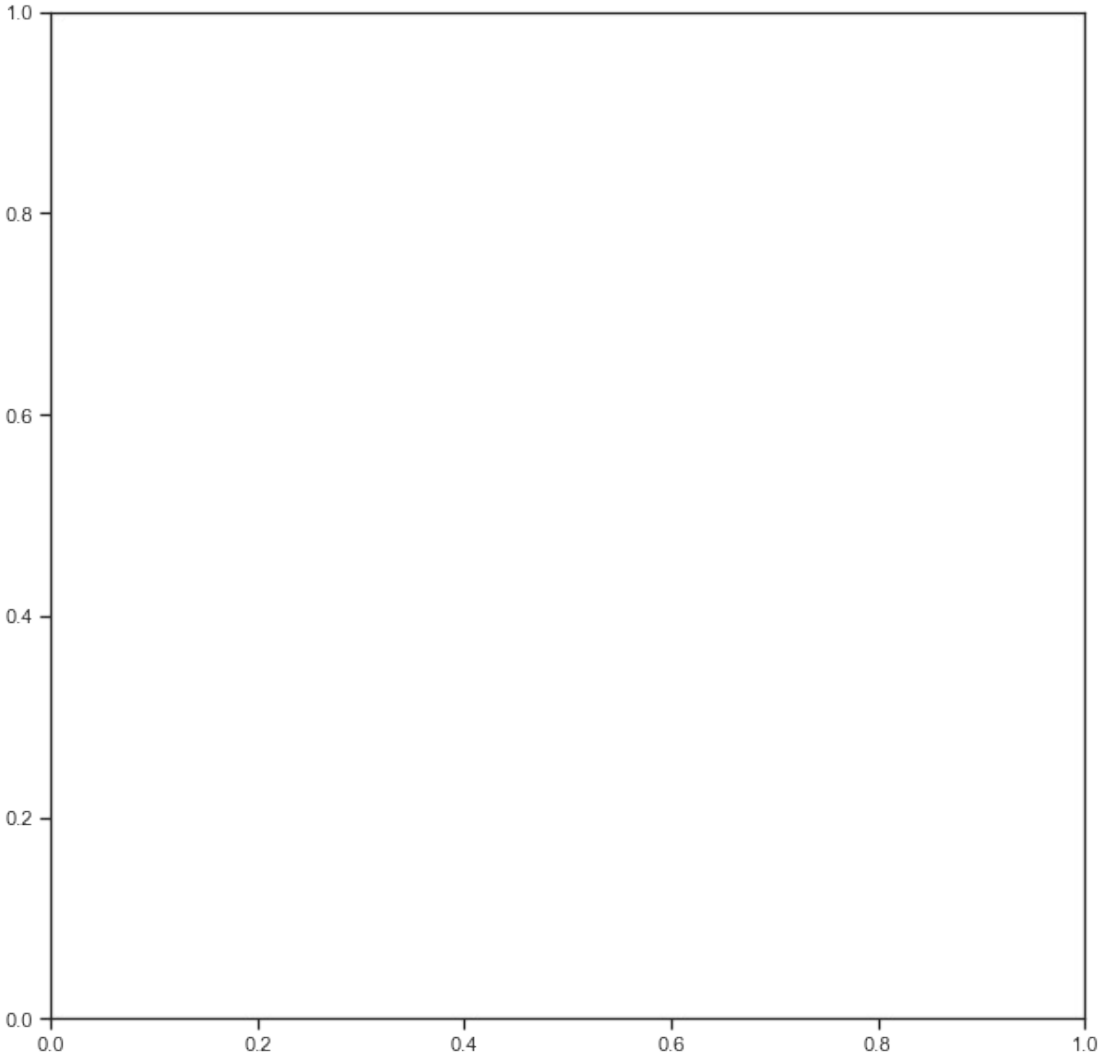
```
def draw(xlabel, ylabel, grid, title, t, rng, bins):
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.grid(grid)
    plt.hist(t, range=rng,bins=bins)
    plt.show()

draw('total_area', 'amount',True ,"Dependence of the number of ads on
the area", df['total_area'], (0,200), 50)
```




```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='total_area', y='amount', data=df)
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-28-f3635995e612> in <module>
      1 fig, ax = plt.subplots(figsize=(10,10))
----> 2 sns.scatterplot(ax=ax, x='total_area', y='amount', data=df)

D:\Anaconda\lib\site-packages\seaborn\relational.py in scatterplot(x,
y, hue, style, size, data, palette, hue_order, hue_norm, sizes,
size_order, size_norm, markers, style_order, x_bins, y_bins, units,
estimator, ci, n_boot, alpha, x_jitter, y_jitter, legend, ax,
**kwargs)
   1333         x_bins=x_bins, y_bins=y_bins,
   1334         estimator=estimator, ci=ci, n_boot=n_boot,
-> 1335         alpha=alpha, x_jitter=x_jitter, y_jitter=y_jitter,
legend=legend,
   1336     )
   1337

D:\Anaconda\lib\site-packages\seaborn\relational.py in __init__(self,
x, y, hue, size, style, data, palette, hue_order, hue_norm, sizes,
size_order, size_norm, dashes, markers, style_order, x_bins, y_bins,
units, estimator, ci, n_boot, alpha, x_jitter, y_jitter, legend)
```

```
    850
    851         plot_data = self.establish_variables(
--> 852             x, y, hue, size, style, units, data
    853         )
    854

D:\Anaconda\lib\site-packages\seaborn\relational.py in
establish_variables(self, x, y, hue, size, style, units, data)
    140                 if isinstance(var, string_types):
    141                     err = "Could not interpret input
'{}'".format(var)
--> 142                     raise ValueError(err)
    143
    144             # Extract variable names

ValueError: Could not interpret input 'amount'
```



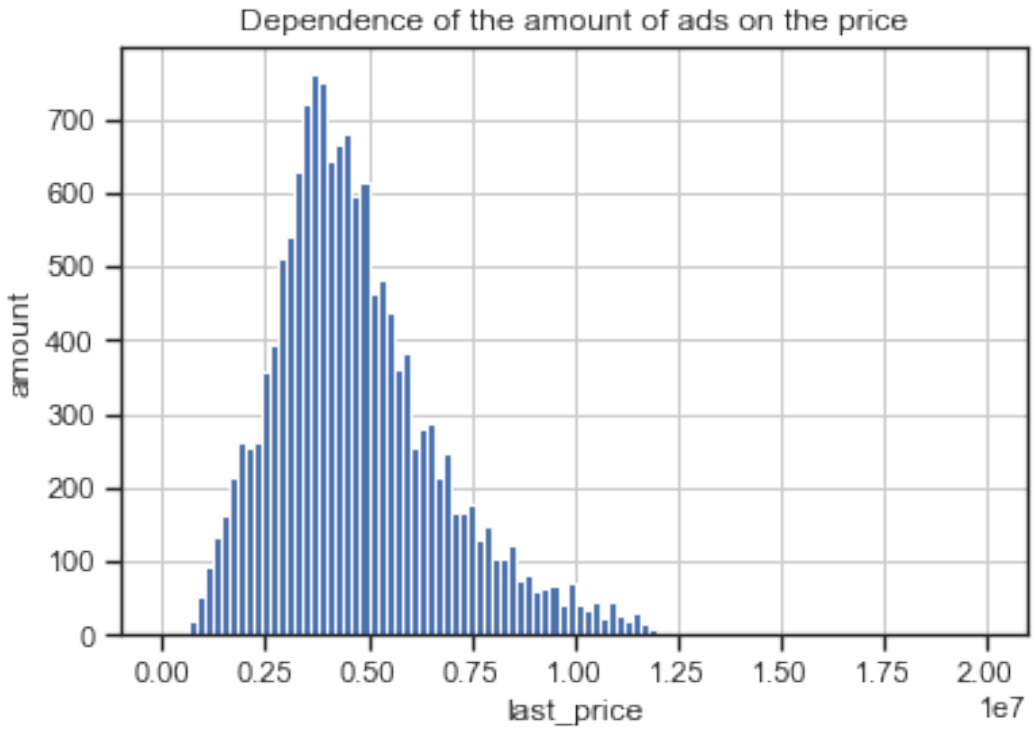

```
df.query('total_area < 60').shape[0] / df.shape[0]
```

```
0.7110897611858359
```

Гистограмма площадей квартир схожа с графиком нормального распределения. Квартиры с площадью <60 м^2 составляют большую часть объявлений (71%)

Визуализируем зависимость числа объявлений от цен.

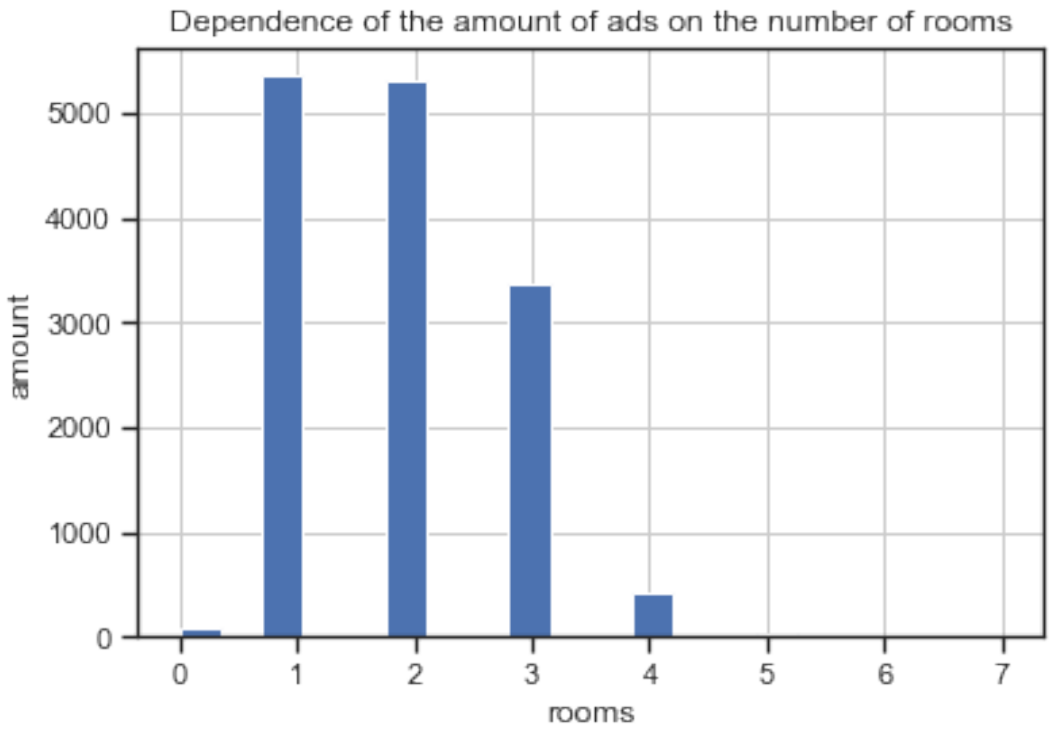
```
draw('last_price','amount',True, "Dependence of the amount of ads on the price",df['last_price'],(0,20000000),100)
```



Гистограмма цен похожа на нормальное распреление.

Визуализируем зависимость числа объявлений от числа комнат.

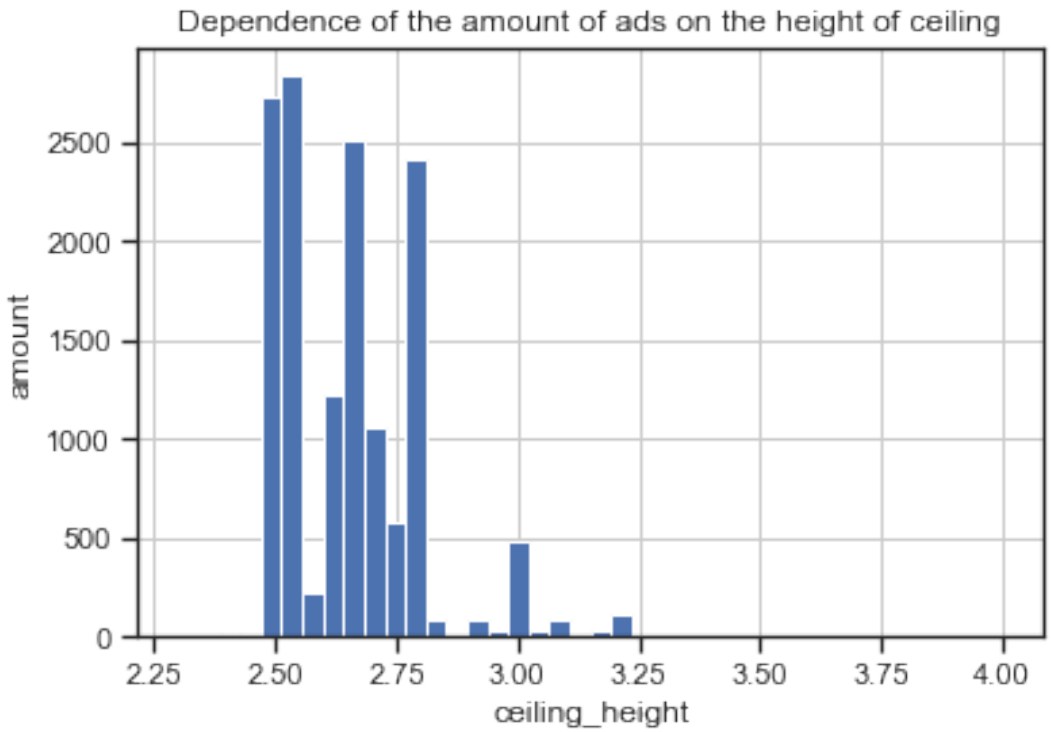
```
draw('rooms','amount',True, "Dependence of the amount of ads on the number of rooms",df['rooms'],(0,7),20)
```



Гистограмма кол-ва комнат представляет собой распределение Пуассона.

Визуализируем зависимость числа объявлений от высоты потолков.

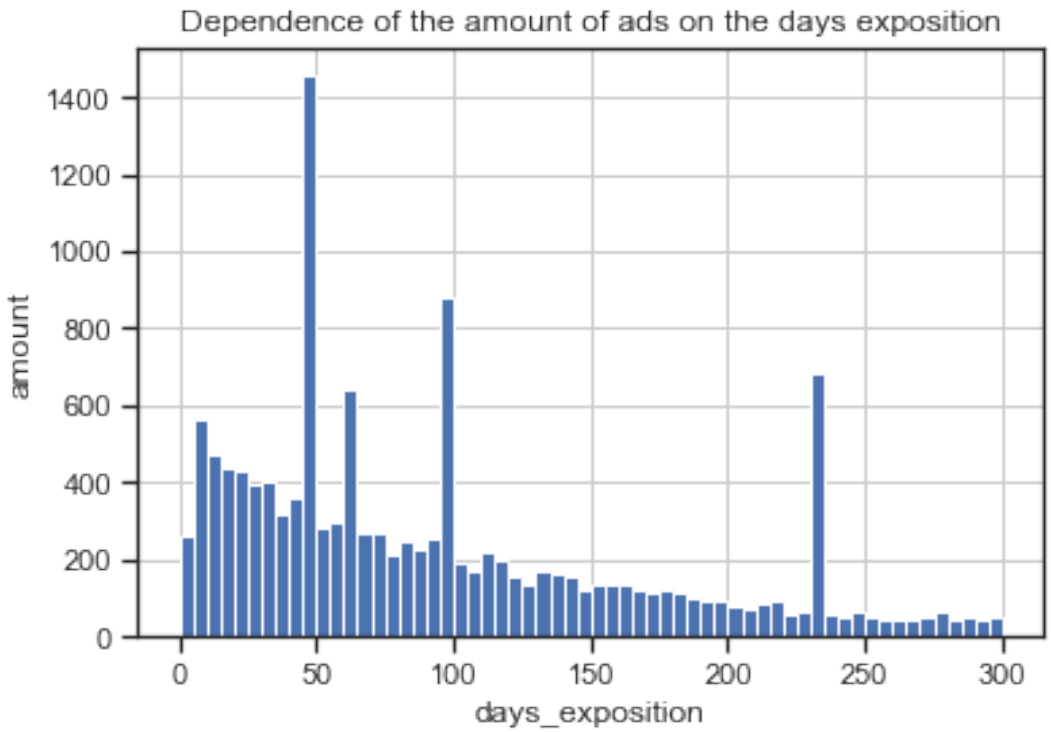
```
draw('ceiling_height','amount',True, "Dependence of the amount of ads
on the height of ceiling",df['ceiling_height'],(2.30,4),40)
```



Изучим время продажи квартиры (к оглавлению)

Построим гистограмму. Посчитаем среднее и медиану. Опишем, сколько обычно занимает продажа. Постараемся понять, когда можно считать, что продажи прошли очень быстро, а когда необычно долго

```
draw('days_exposition','amount',True, "Dependence of the amount of ads
on the days exposition",df['days_exposition'],(0,300),60)
```




```
df['days_exposition'].value_counts()
```

```
45     1239
95      705
231     633
60      374
7       166
       ...
457       1
1         1
509       1
471       1
373       1
Name: days_exposition, Length: 506, dtype: int64
```

График похож на распределение Пуассона.

```
df['days_exposition'].describe()
```

```
count    14572.000000
mean       118.723579
std        108.950272
min          1.000000
25%         44.000000
50%         84.000000
75%        171.000000
max        509.000000
Name: days_exposition, dtype: float64
```

```
print('mean =',df['days_exposition'].describe()[1])
```

```
mean = 118.72357946747186
```

```
print('median =',df['days_exposition'].median())
```

```
median = 84.0
```

Среднее и медианное значения отличаются достаточно сильно.

```
print('Быстро продано если кол-во дней
меньше',df['days_exposition'].describe()[4])
print('Продается долго если кол-во дней
больше',df['days_exposition'].describe()[6])
```

```
Быстро продано если кол-во дней меньше 44.0
Продается долго если кол-во дней больше 171.0
```

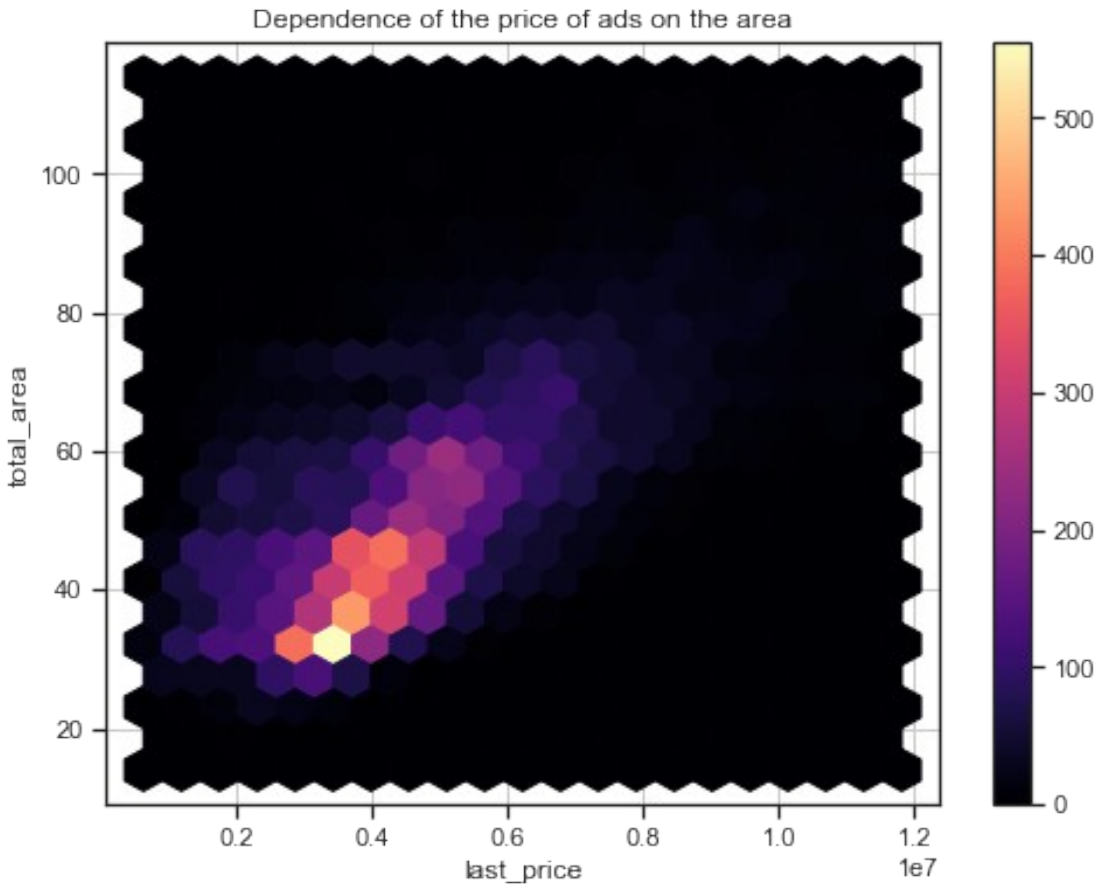
Выясним какие факторы больше всего влияют на стоимость квартиры (к оглавлению)

Изучим, зависит ли цена от квадратного метра, числа комнат, этажа (первого или последнего), удалённости от центра. Также изучим зависимость от даты размещения: дня недели, месяца и года.

Посмотрим на корреляцию стоимости и площади

```
df_2.plot(x='last_price', y='total_area', kind='hexbin', gridsize=20,
          figsize=(8, 6), sharex=False, grid=True,cmap = 'magma',
title="Dependence of the price of ads on the area")
print("Корреляция =", df_2['last_price'].corr(df_2['total_area']))
```

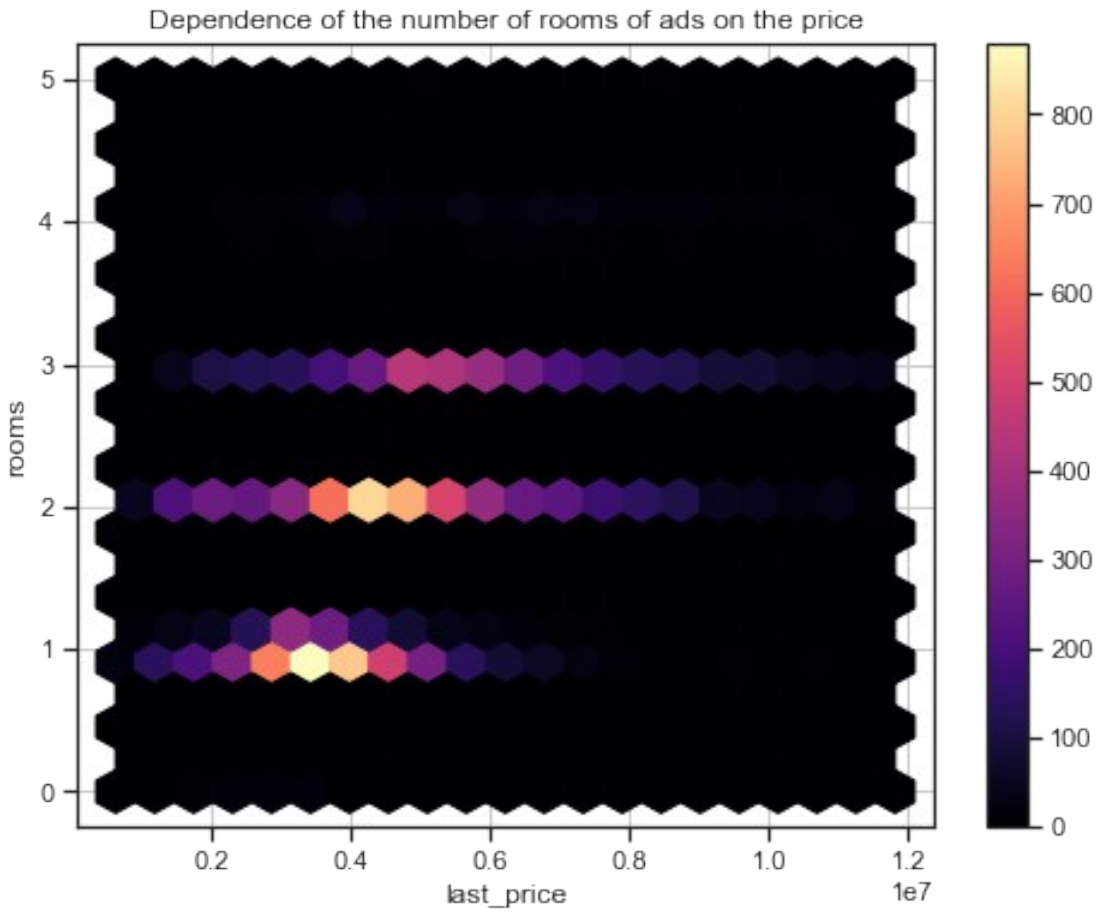
```
Корреляция = 0.6912826867387294
```



Стоимость и число комнат

```
df_2.plot(x='last_price', y='rooms', kind='hexbin', gridsize=20,
figsize=(8, 6), sharex=False,
          grid=True,cmap = 'magma', title="Dependence of the number of
rooms of ads on the price")
print("Корреляция =", df_2['last_price'].corr(df_2['rooms']))
```

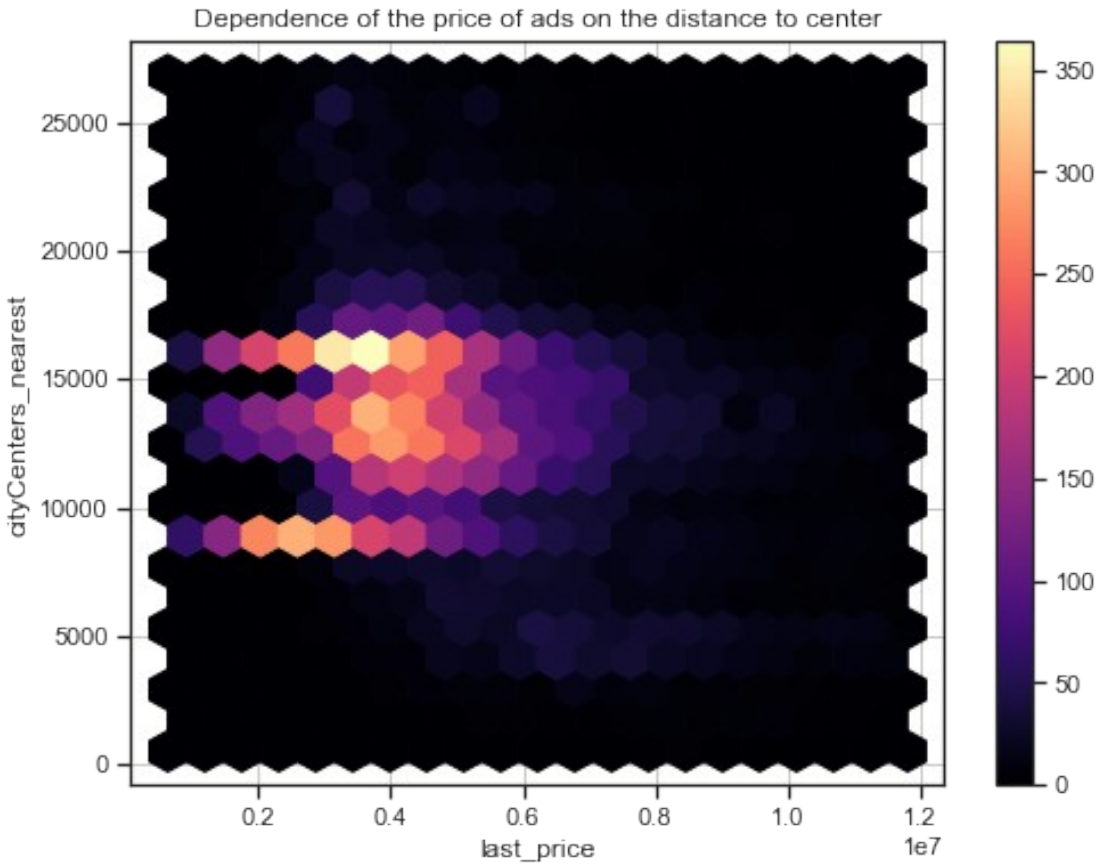
```
Корреляция = 0.4366006219385842
```



Стоимость и расстояние до центра

```
df_2.plot(x='last_price', y='cityCenters_nearest', kind='hexbin',
gridsize=20, figsize=(8, 6), sharex=False,
          grid=True,cmap = 'magma', title="Dependence of the price of
ads on the distance to center")
print("Корреляция =",
df_2['last_price'].corr(df_2['cityCenters_nearest']))
```

```
Корреляция = -0.19130619547949246
```

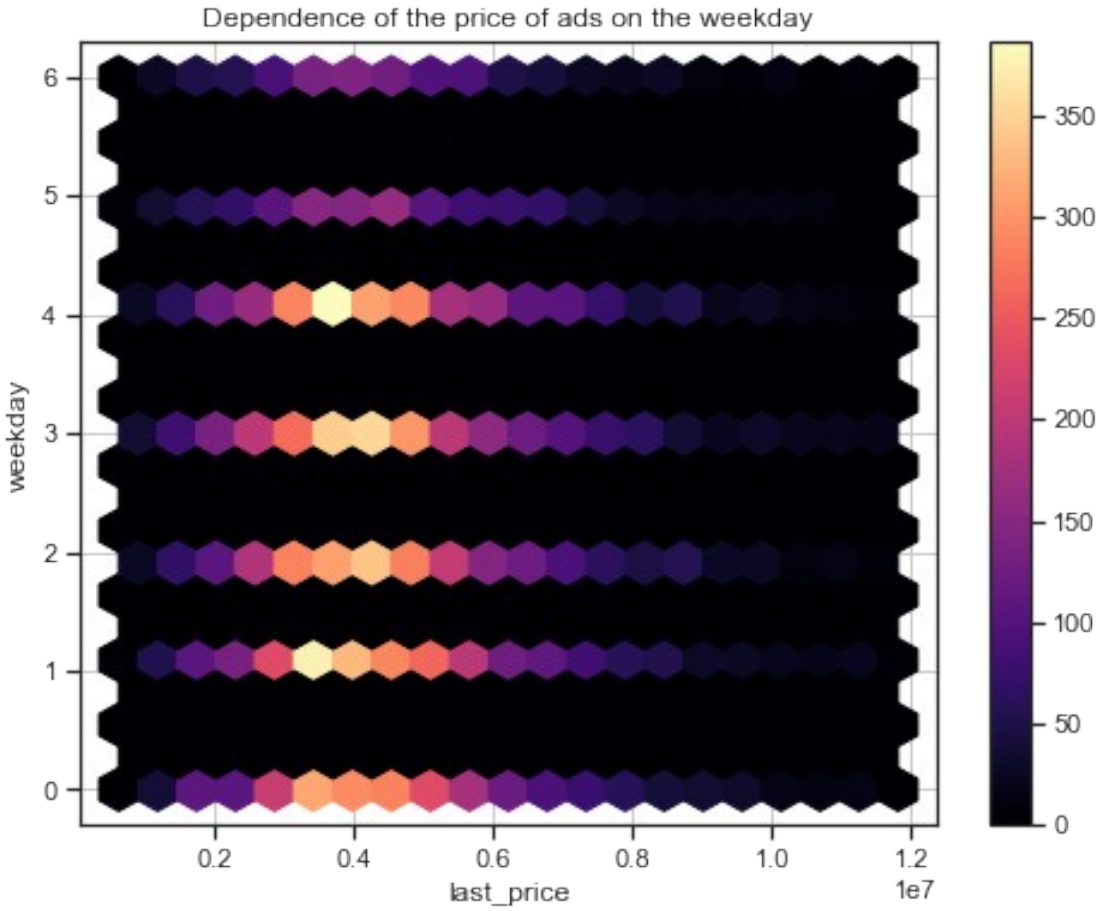


Получили отрицательную корреляцию, что логично, ведь чем меньше расстояние до центра, тем дороже квартира.

Стоимость и день недели

```
df_2.plot(x='last_price', y='weekday', kind='hexbin', gridsize=20,
figsize=(8, 6), sharex=False,
          grid=True,cmap = 'magma', title="Dependence of the price of
ads on the weekday")
print("Корреляция =", df_2['last_price'].corr(df_2['weekday']))
```

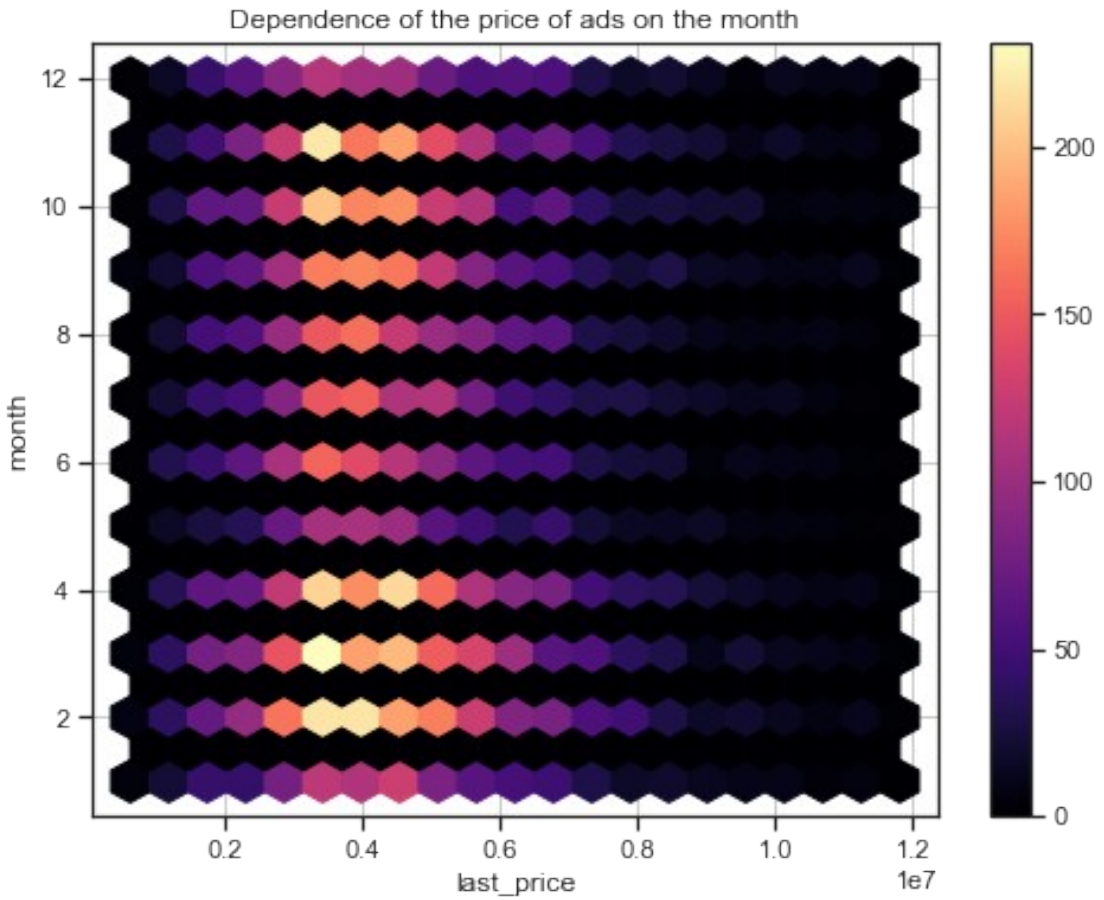
```
Корреляция = -0.011713440558013082
```



Стоимость и месяц

```
df_2.plot(x='last_price', y='month', kind='hexbin', gridsize=20,
figsize=(8, 6), sharex=False,
          grid=True,cmap = 'magma', title="Dependence of the price of
ads on the month")
print("Корреляция =", df_2['last_price'].corr(df_2['month']))
```

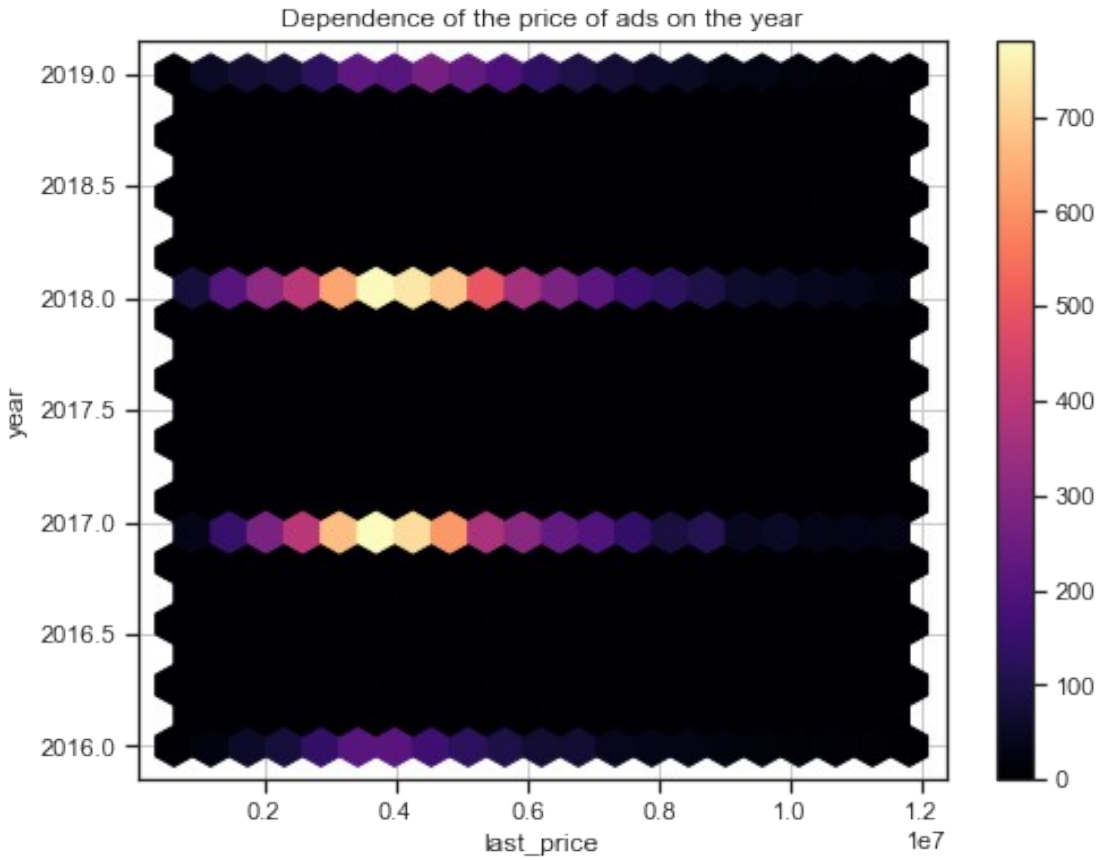
```
Корреляция = 0.003766493258819616
```



Стоимость и год

```
df_2.plot(x='last_price', y='year', kind='hexbin', gridsize=20,
figsize=(8, 6), sharex=False,
          grid=True,cmap = 'magma', title="Dependence of the price of
ads on the year")
print("Корреляция =", df_2['last_price'].corr(df_2['year']))
```

```
Корреляция = 0.038637768240697395
```



Выводы:

- Стоимость квартиры сильно зависит от ее площади, эта зависимость похожа на линейную.
- Стоимость сильно зависит от кол-ва комнат. Объявлений с однокомнатной квартирой стоимостью от 3,5 - 4 млн больше всего.
- Объявления реже выкладывают в субботу и воскресенье.
- Объявления реже выкладывают в декабре, январе, апреле, мае и июне.
- Объявления зачительно реже выкладывали в 2016 году, чем в 2017 и 2018.

Исследуем населенные пункты (к оглавлению)

Выберем 10 населённых пунктов с наибольшим числом объявлений. Посчитаем среднюю цену квадратного метра в этих населённых пунктах и выделим населённые пункты с самой высокой и низкой стоимостью жилья.

```
df_2['locality_name'].value_counts()[0:10]
```

```
Санкт-Петербург      10011
посёлок Мурино         336
Всеволожск             306
Гатчина                242
посёлок Парголово      187
деревня Кудрово        181
Выборг                 177
посёлок Шушары         175
Кудрово                120
Сертолово              118
Name: locality_name, dtype: int64
```

```
top_10 = ['Санкт-Петербург', 'посёлок Мурино', 'Всеволожск',
'Гатчина', 'посёлок Парголово', 'деревня Кудрово', 'Выборг', 'посёлок
Шушары', 'Кудрово','Сертолово']
df_2.query('locality_name in @top_10').groupby('locality_name')
['price_p_m'].mean().sort_values(ascending=False)
```

```
locality_name
Санкт-Петербург      103709.530417
Кудрово              100177.483333
деревня Кудрово       92906.138122
посёлок Парголово     88755.449198
посёлок Мурино        85283.586310
посёлок Шушары        80088.708571
Гатчина               69723.900826
Сертолово             69619.779661
Всеволожск            67125.460784
Выборг                58231.180791
Name: price_p_m, dtype: float64
```

Исследуем центр Санкт-Петербурга (к оглавлению)

Создадим столбец с расстоянием до центра в километрах и округлим до целых значений. Посчитаем среднюю цену для каждого километра.

```
df_piter = df_2.query('locality_name == "Санкт-Петербург"')
df_piter['center_km'] = round(df_piter['cityCenters_nearest'] / 1000)
df_piter_2 = df_piter.groupby('center_km')['last_price'].mean()
df_piter_2
```

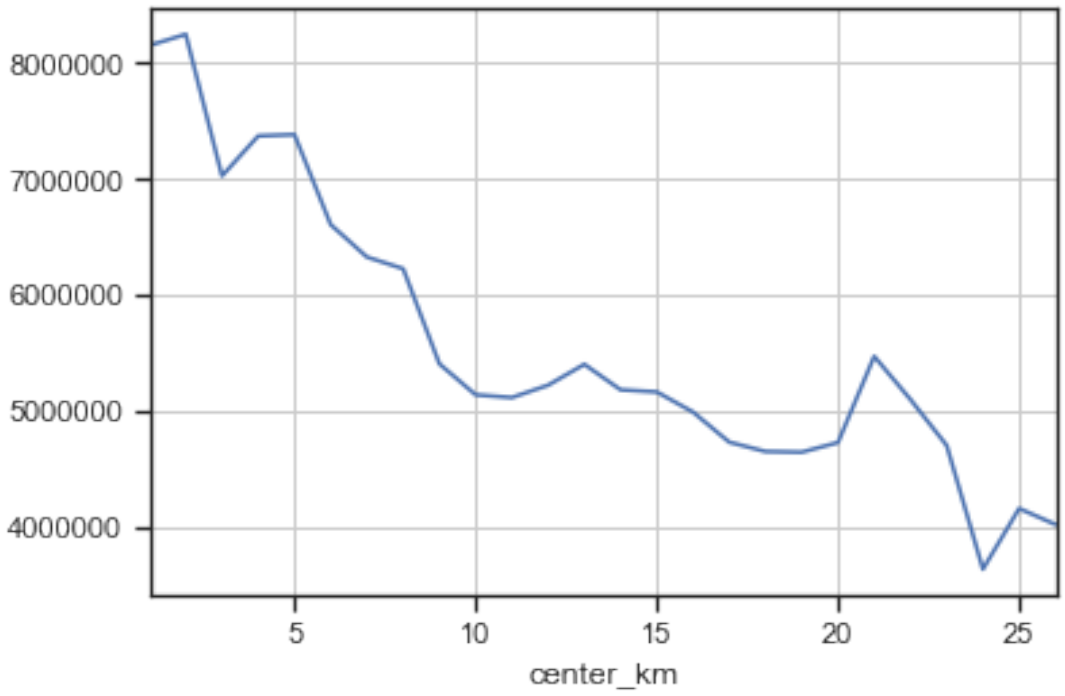
```
center_km
1.0     8.154762e+06
2.0     8.250513e+06
3.0     7.026788e+06
4.0     7.372734e+06
5.0     7.383868e+06
6.0     6.606312e+06
```

```
7.0     6.325969e+06
8.0     6.225011e+06
9.0     5.402991e+06
10.0    5.135490e+06
11.0    5.111727e+06
12.0    5.218358e+06
13.0    5.399069e+06
14.0    5.178831e+06
15.0    5.160406e+06
16.0    4.986411e+06
17.0    4.726291e+06
18.0    4.645701e+06
19.0    4.640078e+06
20.0    4.722941e+06
21.0    5.468050e+06
22.0    5.095196e+06
23.0    4.698141e+06
24.0    3.629294e+06
25.0    4.153269e+06
26.0    4.015105e+06
Name: last_price, dtype: float64
```

Визуализируем зависимость цены от удаленности от центра.

```
df_piter_2.plot(grid = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23b8517d4c8>
```

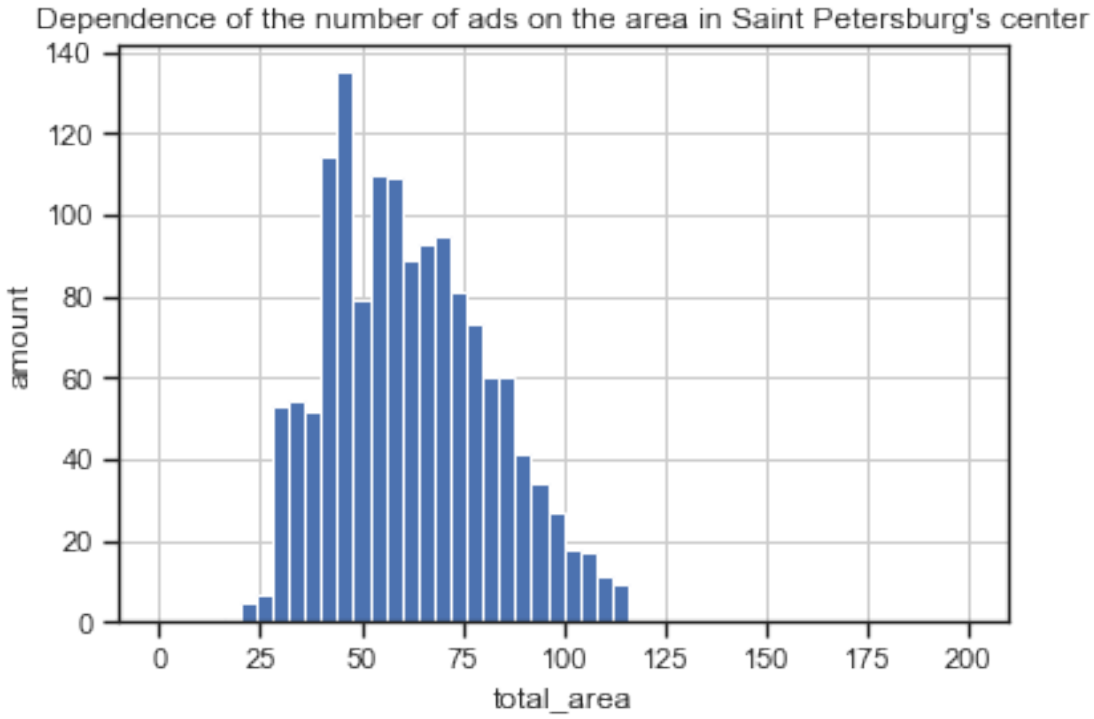


Судя по графику делаем вывод, что центр кончается на 8-ом километре.

Далее проанализируем центр Санкт-Петербурга, рассмотрим параметры, которые влияют на количество объявлений и стоимость квартир в этой области. Сравним их с общими выводами для всех городов.

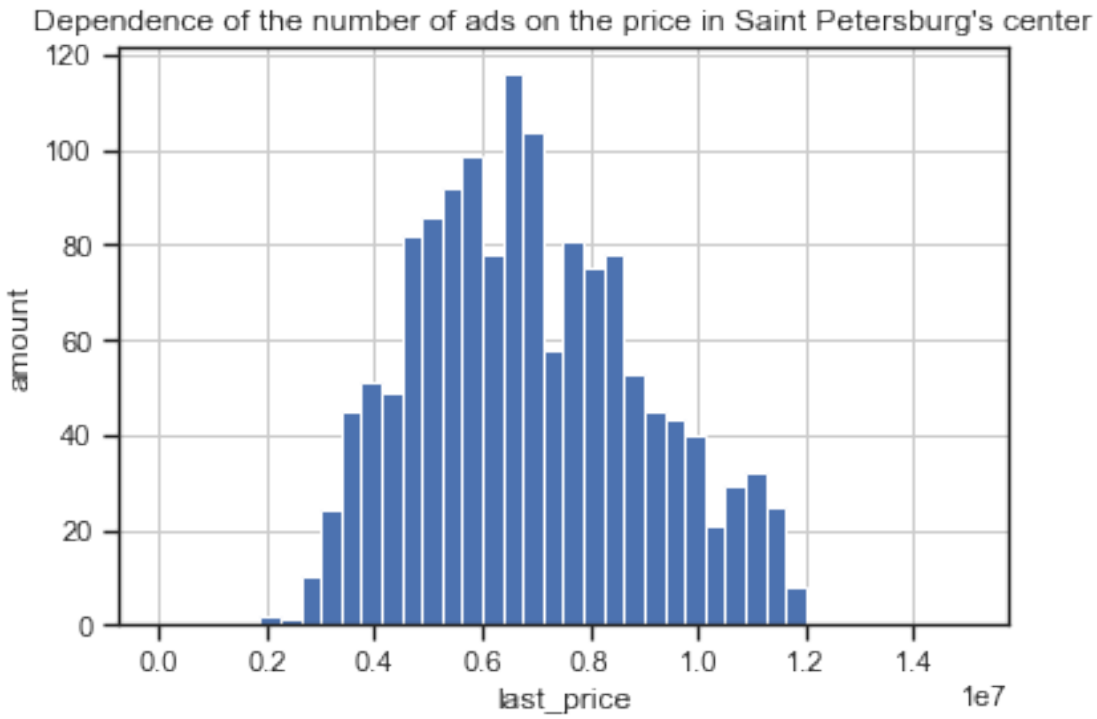
Построим зависимость числа объявлений от площади квартиры.

```
df_piter_center = df_piter.query('center_km < 9')
draw('total_area', 'amount',True ,"Dependence of the number of ads on
the area in Saint Petersburg's center", df_piter_center['total_area'],
(0,200),50)
```



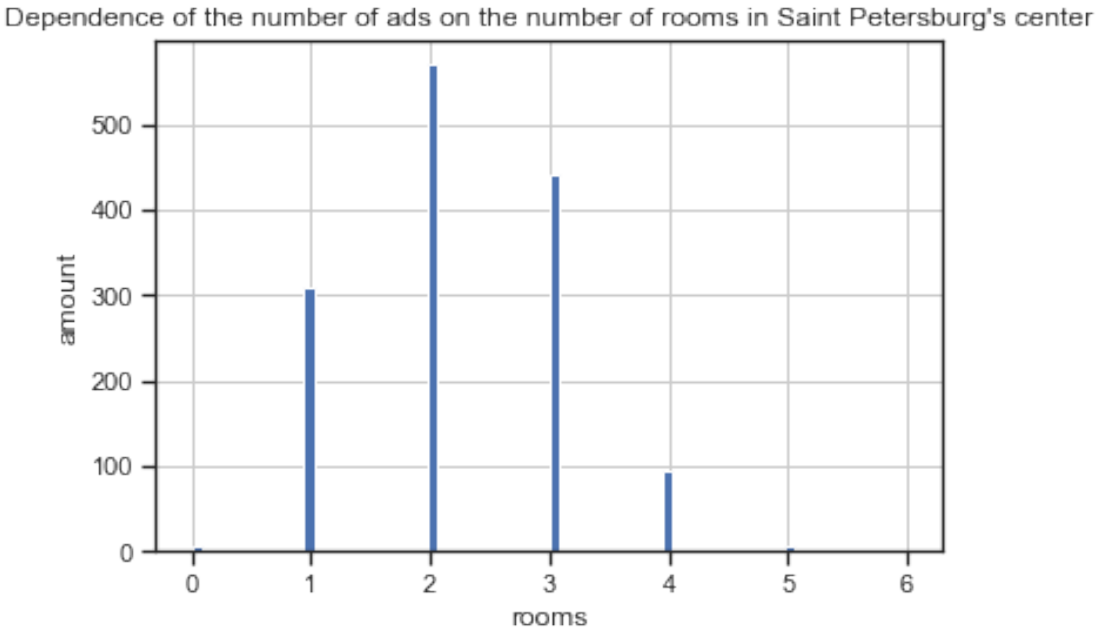
Зависимость числа объявлений от стоимости квартиры.

```
draw('last_price', 'amount',True ,"Dependence of the number of ads on
the price in Saint Petersburg's center",
df_piter_center['last_price'], (0,15000000),40)
```



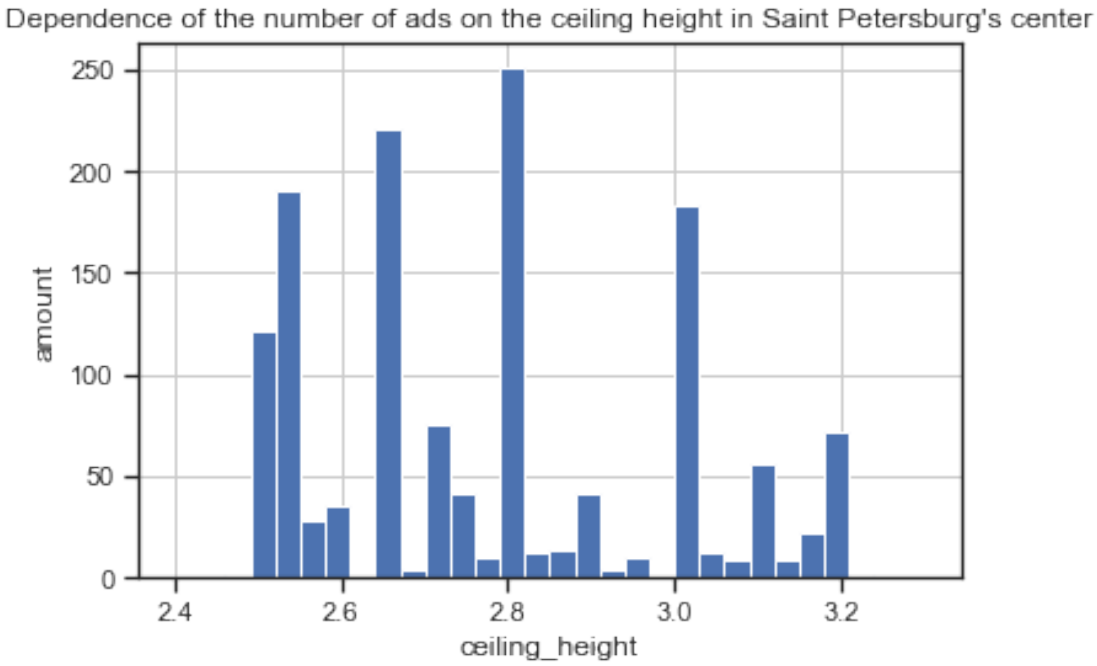
Зависимость числа объявлений от количества комнат в квартире.

```
draw('rooms', 'amount',True ,"Dependence of the number of ads on the
number of rooms in Saint Petersburg's center",
df_piter_center['rooms'], (0,6),70)
```



Зависимость числа объявлений от высоты потолков в квартире.

```
draw('ceiling_height', 'amount',True ,"Dependence of the number of ads
on the ceiling height in Saint Petersburg's center",
df_piter_center['ceiling_height'], (2.4,3.3),30)
```

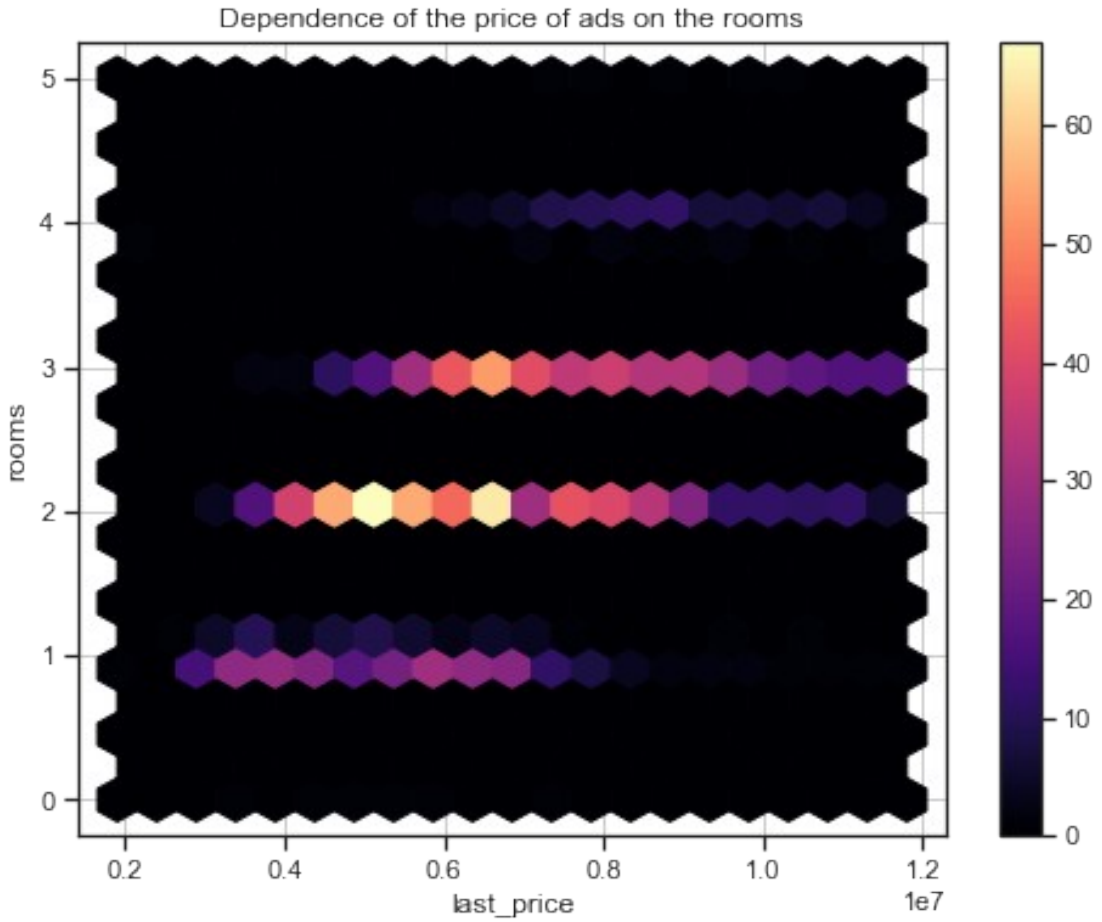


Покажем далее какие параметры влияют на стоимость квартир в Санкт-Петербурге.

Стоимость и число комнат.

```
df_piter_center.plot(x='last_price', y='rooms', kind='hexbin',
gridsize=20,
          figsize=(8, 6), sharex=False, grid=True,cmap = 'magma',
title="Dependence of the price of ads on the rooms")
print("Корреляция =",
df_piter_center['last_price'].corr(df_piter_center['rooms']))
```

```
Корреляция = 0.49018698492430296
```

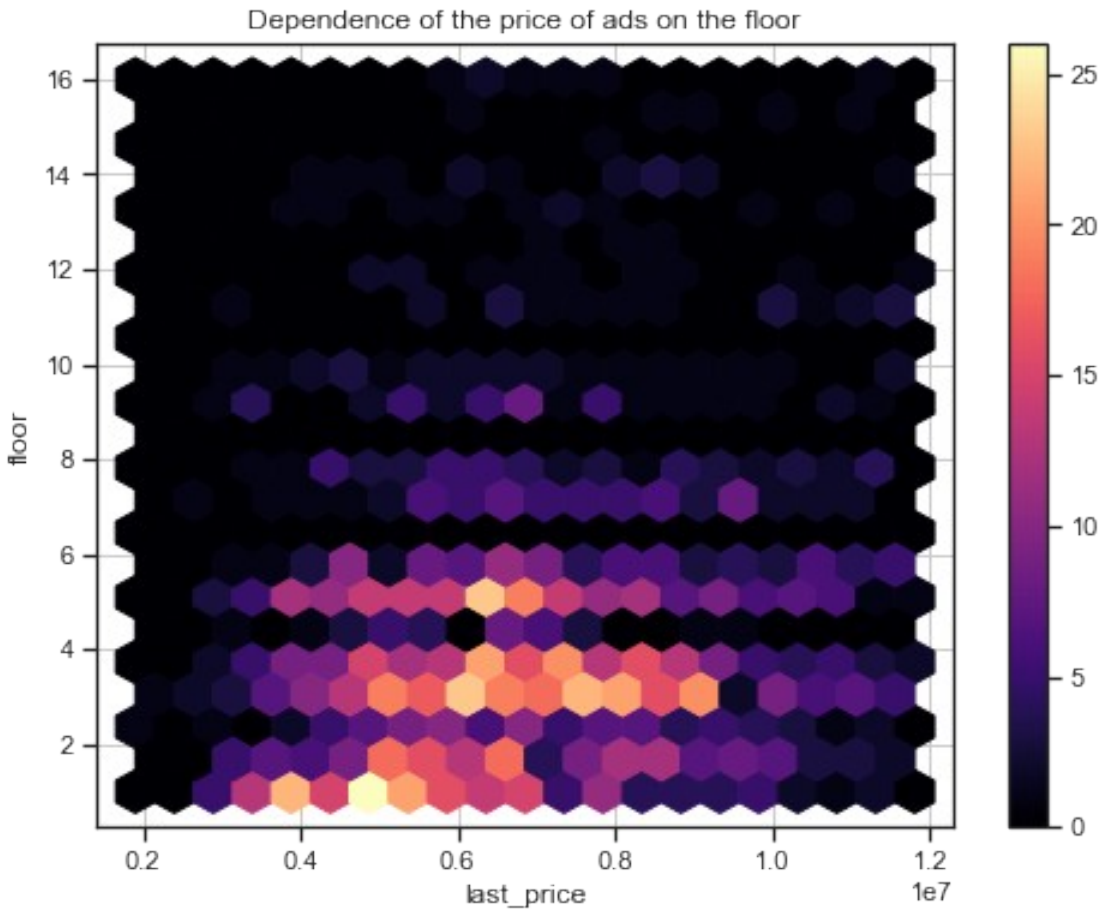


Корелляция = 1/2 - зависимость есть.

Стоимость и этаж.

```
df_piter_center.plot(x='last_price', y='floor', kind='hexbin',
gridsize=20,
          figsize=(8, 6), sharex=False, grid=True,cmap = 'magma',
title="Dependence of the price of ads on the floor")
print("Корреляция =",
df_piter_center['last_price'].corr(df_piter_center['floor']))
```

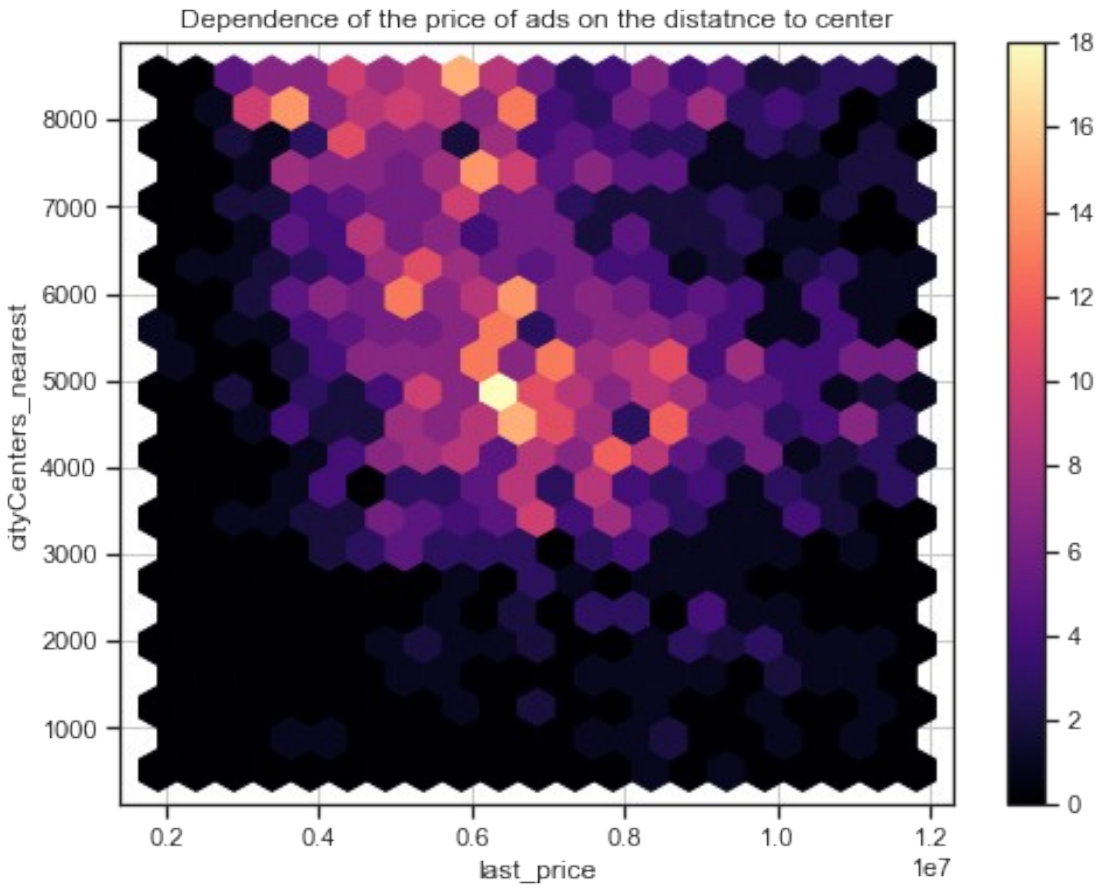
```
Корреляция = 0.1598238132011353
```



Стоимость и расстояние до центра.

```
df_piter_center.plot(x='last_price', y='cityCenters_nearest',
kind='hexbin', gridsize=20,
          figsize=(8, 6), sharex=False, grid=True,cmap = 'magma',
title="Dependence of the price of ads on the distatnce to center")
print("Корреляция =",
df_piter_center['last_price'].corr(df_piter_center['cityCenters_neares
t']))
```

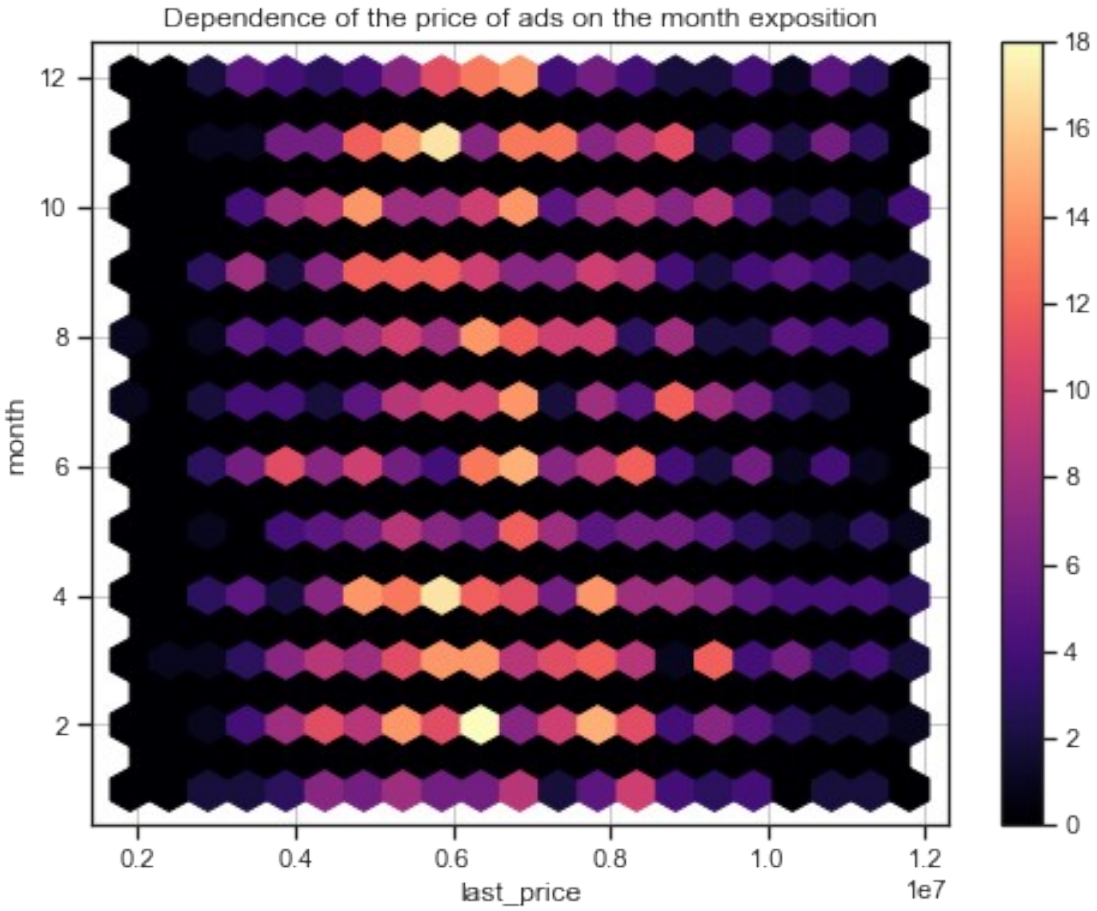
```
Корреляция = -0.23869913166569612
```



Стоимость и месяц.

```
df_piter_center.plot(x='last_price', y='month', kind='hexbin',
gridsize=20,
          figsize=(8, 6), sharex=False, grid=True,cmap = 'magma',
title="Dependence of the price of ads on the month exposition")
print("Корреляция =",
df_piter_center['last_price'].corr(df_piter_center['month']))
```

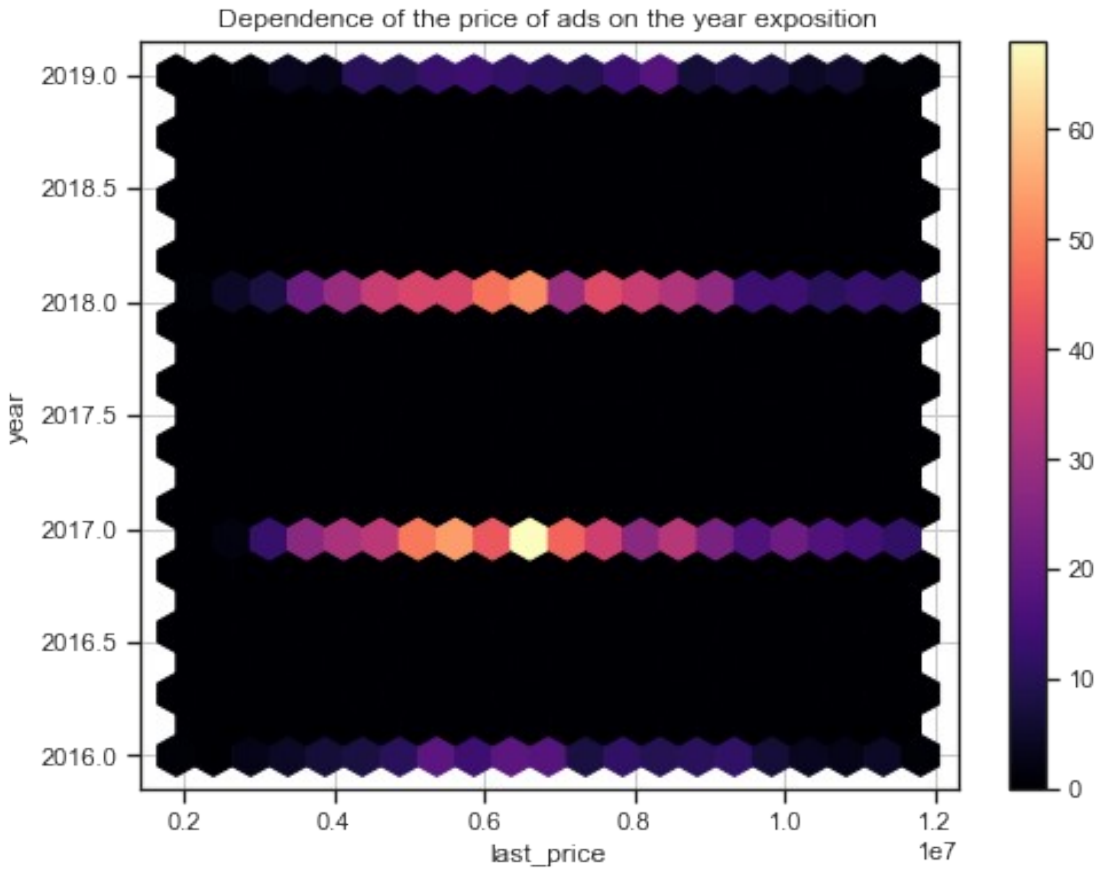
```
Корреляция = -0.0011249609015818716
```



Стоимость и год.

```
df_piter_center.plot(x='last_price', y='year', kind='hexbin',
gridsize=20,
          figsize=(8, 6), sharex=False, grid=True,cmap = 'magma',
title="Dependence of the price of ads on the year exposition")
print("Корреляция =",
df_piter_center['last_price'].corr(df_piter_center['year']))
```

```
Корреляция = 0.026038139843545978
```



Вывод:

- Стоимость сильно зависит от количества комнат. Корелляция = 1/2.
- Стоимость сильно зависит от площади квартиры.
- Объявления реже выкладывают в субботу и воскресенье.
- Объявления реже выкладывают в декабре, январе, апреле, мае и июне.
- Объявления зачительно реже выкладывали в 2016 году, чем в 2017 и 2018.

Такой-же результат мы получили, анализируя весь город.

Общий вывод (к оглавлению)

В ходе этого исследования данные были предобработаны, выбивающиеся значения были обнаружены и отсечены, также были очищены неинформативные и пустые данные.

Были определены самые дорогие населенные пункты и оценены зависимости цены от многих факторов, в том числе площадь, количество комнат, дата публикаци объявления и другие.

Определен радиус ценового "центра" Санкт-Петербурга.

По окончании исследования, можем принять все три поставленные гипотезы.