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

Лабораторная работа №3

«Обработка признаков часть 2»

ИСПОЛНИТЕЛЬ:

Федорова Антонина Алексеевна
Группа ИУ5-24М

_____

"__"_____2021 г.

Москва 2021

Целью работы является: изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

Задание:

Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

- масштабирование признаков (не менее чем тремя способами);
- обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
- обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);
- отбор признаков:
    - один метод из группы методов фильтрации (filter methods);
    - один метод из группы методов обертывания (wrapper methods);
    - один метод из группы методов вложений (embedded methods).

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
import scipy.stats as stats
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
```

Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

- масштабирование признаков (не менее чем тремя способами);
- обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
- обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);
- отбор признаков:
  - один метод из группы методов фильтрации (filter methods);
  - один метод из группы методов обертывания (wrapper methods);
  - один метод из группы методов вложений (embedded methods).

In [2]:
```
data = pd.read_csv('/Users/a.fedorova/Desktop/учеба/Великолепная ма
```

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

Out[3]:

artists	danceability	energy	key	loudness	mode	speechiness	acousticness	instrum
Drake	0.754	0.449	7.0	-9.211	1.0	0.1090	0.0332	
XTENTACION	0.740	0.613	8.0	-4.880	1.0	0.1450	0.2580	
Post Malone	0.587	0.535	5.0	-6.090	0.0	0.0898	0.1170	
Post Malone	0.739	0.559	8.0	-8.011	1.0	0.1170	0.5800	
Drake	0.835	0.626	1.0	-5.833	1.0	0.1250	0.0589	

In [4]:
```
data.describe()
```

Out[4]:

	danceability	energy	key	loudness	mode	speechiness	acoustici
count	100.00000	100.000000	100.000000	100.000000	100.000000	100.000000	100.00(
mean	0.71646	0.659060	5.330000	-5.677640	0.590000	0.115569	0.19
std	0.13107	0.145067	3.676447	1.777577	0.494311	0.104527	0.22(
min	0.25800	0.296000	0.000000	-10.109000	0.000000	0.023200	0.00(
25%	0.63550	0.562000	1.750000	-6.650500	0.000000	0.045350	0.04(
50%	0.73300	0.678000	5.000000	-5.566500	1.000000	0.074950	0.10
75%	0.79825	0.772250	8.250000	-4.363750	1.000000	0.137000	0.24
max	0.96400	0.909000	11.000000	-2.384000	1.000000	0.530000	0.93

In [5]:
```
X_ALL = data.drop(['id', 'name', 'artists', 'liveness'], axis=1)
```

In [6]:
```
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=X_ALL.columns)
    return res
```

Масштабирование признаков

StandardScaler

In [7]:
```
# Обучаем StandardScaler на всей выборке и масштабируем
cs11 = StandardScaler()
data_cs11_scaled_temp = cs11.fit_transform(X_ALL)
# формируем DataFrame на основе массива
data_cs11_scaled =  arr_to_df(data_cs11_scaled_temp)
data_cs11_scaled
```

Out[7]:

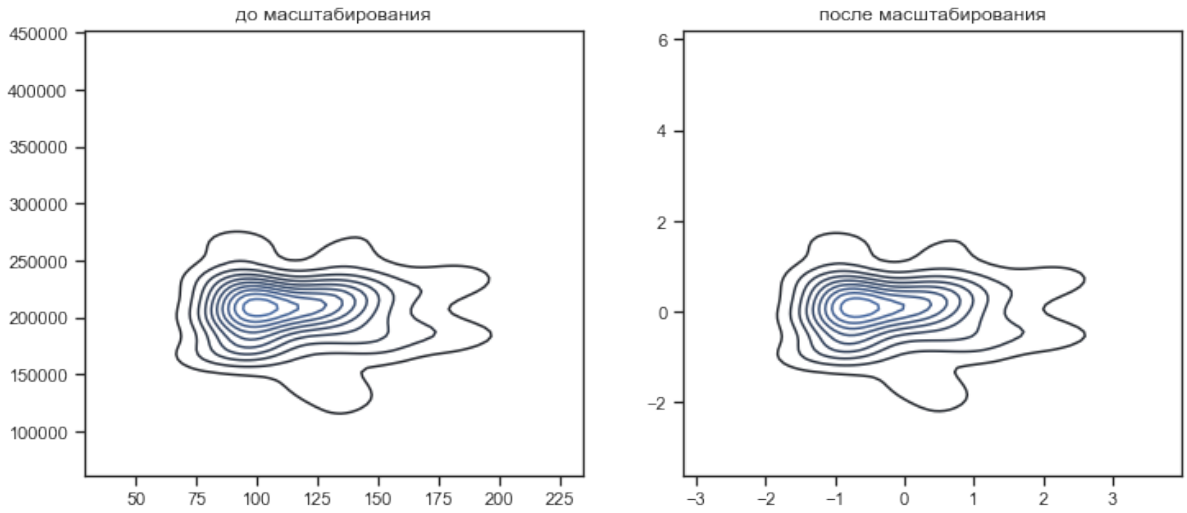
	danceability	energy	key	loudness	mode	speechiness	acousticness	ins
0	0.287854	-1.455314	0.456531	-1.997753	0.833616	-0.063162	-0.739184	
1	0.180503	-0.319108	0.729903	0.450984	0.833616	0.282982	0.283383	
2	-0.992691	-0.859498	-0.090213	-0.233147	-1.199593	-0.247772	-0.357995	
3	0.172835	-0.693224	0.729903	-1.319276	0.833616	0.013759	1.748092	
4	0.908957	-0.229043	-1.183701	-0.087840	0.833616	0.090680	-0.622280	
...	...	...	...	...	...	...	...	
95	-0.248901	-0.277539	1.276647	-0.750486	0.833616	-0.740065	-0.564510	
96	-0.601626	-0.007344	-0.636957	-0.362058	0.833616	-0.672759	0.224249	
97	-1.261069	0.699321	-1.457073	0.549363	0.833616	2.177157	-0.556777	
98	-3.515443	-1.538451	1.550019	-0.517542	-1.199593	-0.736219	-0.430776	
99	-1.253401	-0.506166	1.550019	-0.362623	0.833616	-0.763141	2.280300	

100 rows × 12 columns

In [8]:
```
# Построение плотности распределения
def draw_kde(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 5))
    # первый график
    ax1.set_title(label1)
    sns.kdeplot(data=df1[col_list], ax=ax1)
    # второй график
    ax2.set_title(label2)
    sns.kdeplot(data=df2[col_list], ax=ax2)
    plt.show()
```

In [9]:
```
draw_kde(['tempo', 'duration_ms'], data, data_cs11_scaled, 'до масш
```

```
/Users/a.fedorova/opt/anaconda3/lib/python3.7/site-packages/seabor
n/distributions.py:679: UserWarning: Passing a 2D dataset for a bi
variate plot is deprecated in favor of kdeplot(x, y), and it will
cause an error in future versions. Please update your code.
  warnings.warn(warn_msg, UserWarning)
```



Масштабирование "Mean Normalisation"

In [10]:
```
# Разделим выборку на обучающую и тестовую
X_train, X_test, y_train, y_test = train_test_split(X_ALL, data['li
                                                    test_size=0.2,
                                                    random_state=1)

# Преобразуем массивы в DataFrame
X_train_df = arr_to_df(X_train)
X_test_df = arr_to_df(X_test)

X_train_df.shape, X_test_df.shape
```

Out[10]:
```
((80, 12), (20, 12))
```

In [11]:
```
class MeanNormalisation:

    def fit(self, param_df):
        self.means = X_train.mean(axis=0)
        maxs = X_train.max(axis=0)
        mins = X_train.min(axis=0)
        self.ranges = maxs - mins

    def transform(self, param_df):
        param_df_scaled = (param_df - self.means) / self.ranges
        return param_df_scaled

    def fit_transform(self, param_df):
        self.fit(param_df)
        return self.transform(param_df)
```

In [12]:
```
sc21 = MeanNormalisation()
data_cs21_scaled = sc21.fit_transform(X_ALL)
data_cs21_scaled.describe()
```

Out[12]:

	danceability	energy	key	loudness	mode	speechiness	acousticnes
ount	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.00000
ıean	-0.016824	-0.006161	-0.030227	-0.002058	0.002500	-0.007142	0.00878
std	0.185652	0.241376	0.334222	0.231335	0.494311	0.206249	0.25051
min	-0.666200	-0.610254	-0.514773	-0.578758	-0.587500	-0.189402	-0.21278
25%	-0.131498	-0.167658	-0.355682	-0.128667	-0.587500	-0.145696	-0.16749
50%	0.006604	0.025354	-0.060227	0.012406	0.412500	-0.087290	-0.08951
75%	0.099026	0.182176	0.235227	0.168932	0.412500	0.035145	0.06780
max	0.333800	0.409713	0.485227	0.426578	0.412500	0.810598	0.84590

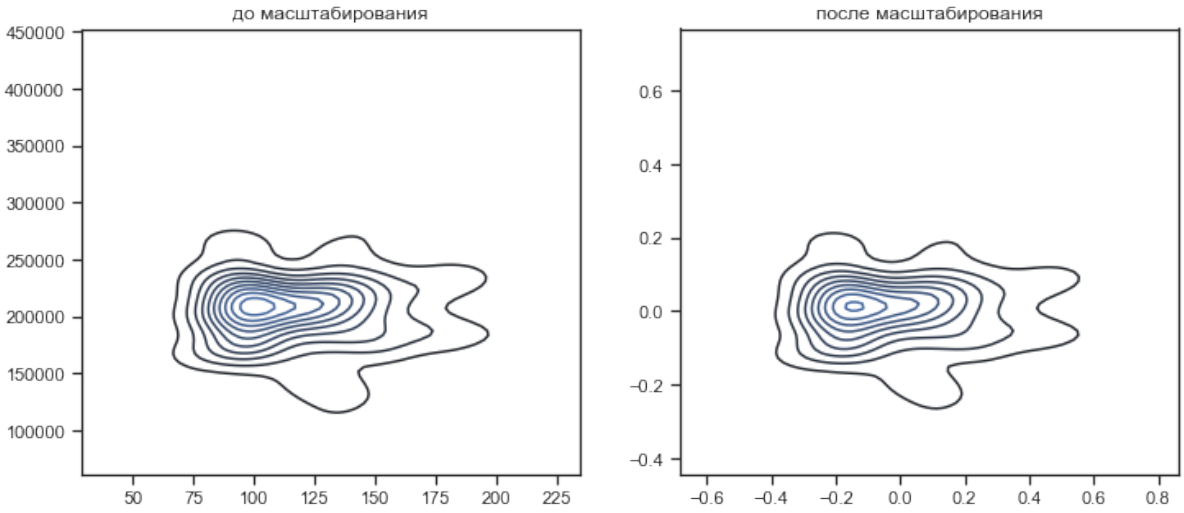
In [13]:
```
cs22 = MeanNormalisation()
cs22.fit(X_train)
data_cs22_scaled_train = cs22.transform(X_train)
data_cs22_scaled_test = cs22.transform(X_test)
```

In [14]:
```
data_cs22_scaled_train.describe()
```

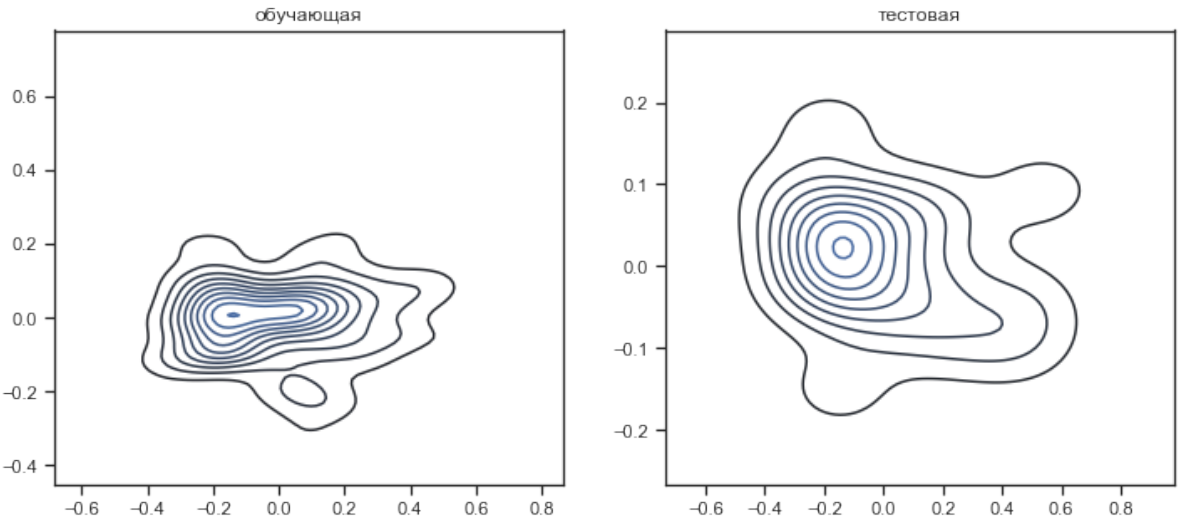
Out[14]:

	danceability	energy	key	loudness	mode	speechiness
count	8.000000e+01	8.000000e+01	8.000000e+01	8.000000e+01	8.000000e+01	8.000000e+01
mean	-1.161181e-16	-2.223915e-16	1.457168e-17	-7.736867e-17	1.942890e-17	4.128642e-17
std	1.742082e-01	2.318961e-01	3.348639e-01	2.281019e-01	4.953901e-01	2.164278e-01
min	-6.662004e-01	-5.902870e-01	-5.147727e-01	-5.734220e-01	-5.875000e-01	-1.894016e-01
25%	-1.106055e-01	-1.622504e-01	-3.329545e-01	-1.217367e-01	-5.875000e-01	-1.440188e-01
50%	1.226983e-02	2.202579e-02	3.068182e-02	-4.187272e-03	4.125000e-01	-7.860843e-02
75%	1.007967e-01	1.701123e-01	3.034091e-01	1.692575e-01	4.125000e-01	2.823846e-02
max	3.337996e-01	4.097130e-01	4.852273e-01	4.265780e-01	4.125000e-01	8.105984e-01

In [15]:
```
draw_kde(['tempo', 'duration_ms'], data, data_cs21_scaled, 'до масш
```



In [16]:
```
draw_kde(['tempo', 'duration_ms'], data_cs22_scaled_train, data_cs2
```



MinMax-масштабирование

In [17]:
```
# Обучаем StandardScaler на всей выборке и масштабируем
cs31 = MinMaxScaler()
data_cs31_scaled_temp = cs31.fit_transform(X_ALL)
# формируем DataFrame на основе массива
data_cs31_scaled = arr_to_df(data_cs31_scaled_temp)
data_cs31_scaled.describe()
```

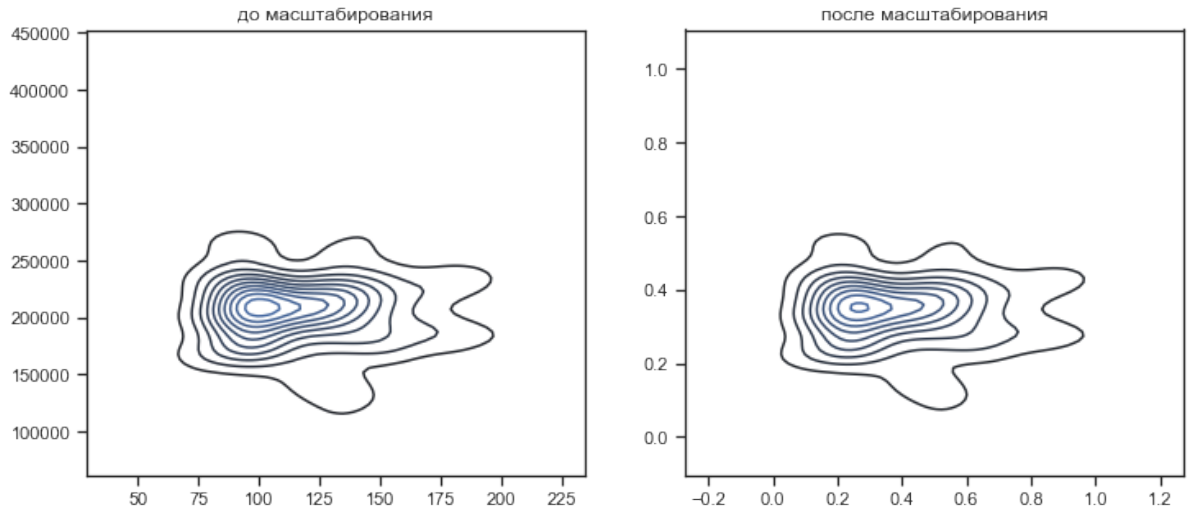
Out[17]:

	danceability	energy	key	loudness	mode	speechiness	acoustici
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.00(
mean	0.649377	0.592268	0.484545	0.573639	0.590000	0.182259	0.20
std	0.185652	0.236651	0.334222	0.230107	0.494311	0.206249	0.23(
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
25%	0.534703	0.433931	0.159091	0.447702	0.000000	0.043706	0.04
50%	0.672805	0.623165	0.454545	0.588026	1.000000	0.102111	0.11(
75%	0.765227	0.776917	0.750000	0.743722	1.000000	0.224546	0.26
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00(

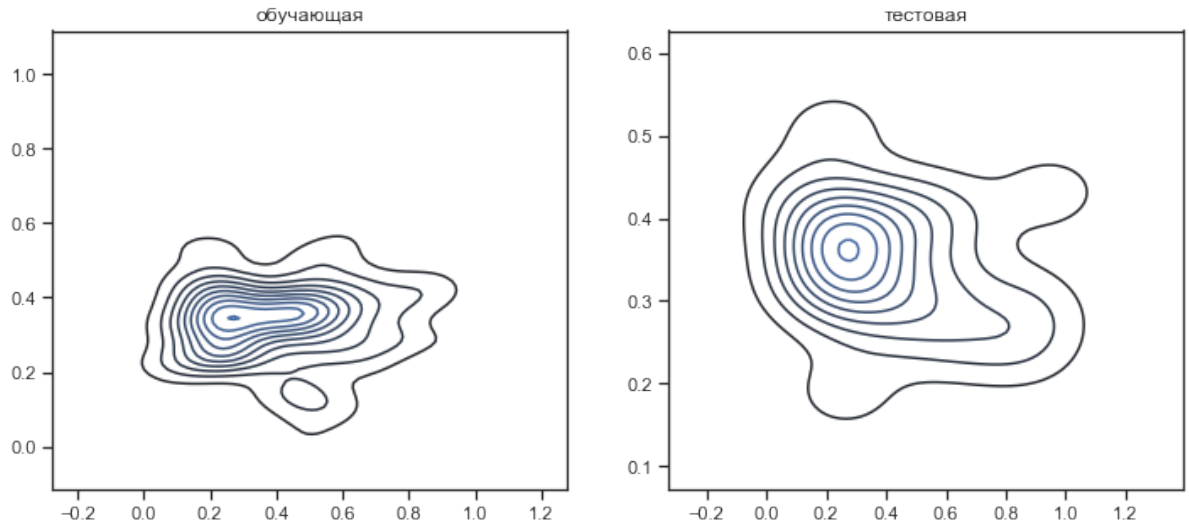
In [18]:
```
cs32 = MinMaxScaler()
cs32.fit(X_train)
data_cs32_scaled_train_temp = cs32.transform(X_train)
data_cs32_scaled_test_temp = cs32.transform(X_test)
# формируем DataFrame на основе массива
data_cs32_scaled_train = arr_to_df(data_cs32_scaled_train_temp)
data_cs32_scaled_test = arr_to_df(data_cs32_scaled_test_temp)
```

In [19]:
```
draw_kde(['tempo', 'duration_ms'], data, data_cs31_scaled, 'до масш
```

```
/Users/a.fedorova/opt/anaconda3/lib/python3.7/site-packages/seabor
n/distributions.py:679: UserWarning: Passing a 2D dataset for a bi
variate plot is deprecated in favor of kdeplot(x, y), and it will
cause an error in future versions. Please update your code.
  warnings.warn(warn_msg, UserWarning)
```



In [20]:
```
draw_kde(['tempo', 'duration_ms'], data_cs32_scaled_train, data_cs3
```



Обработка выбросов для числовых признаков

In [21]:
```
data2 = pd.read_csv('/Users/a.fedorova/Desktop/учеба/Великолепная м
```

In [22]:
```
data2.head()
```

Out[22]:

ood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review
gton	40.64749	-73.97237	Private room	149	1	9	2018-10-19
own	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21
lem	40.80902	-73.94190	Private room	150	3	0	NaN
Hill	40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-05
lem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19

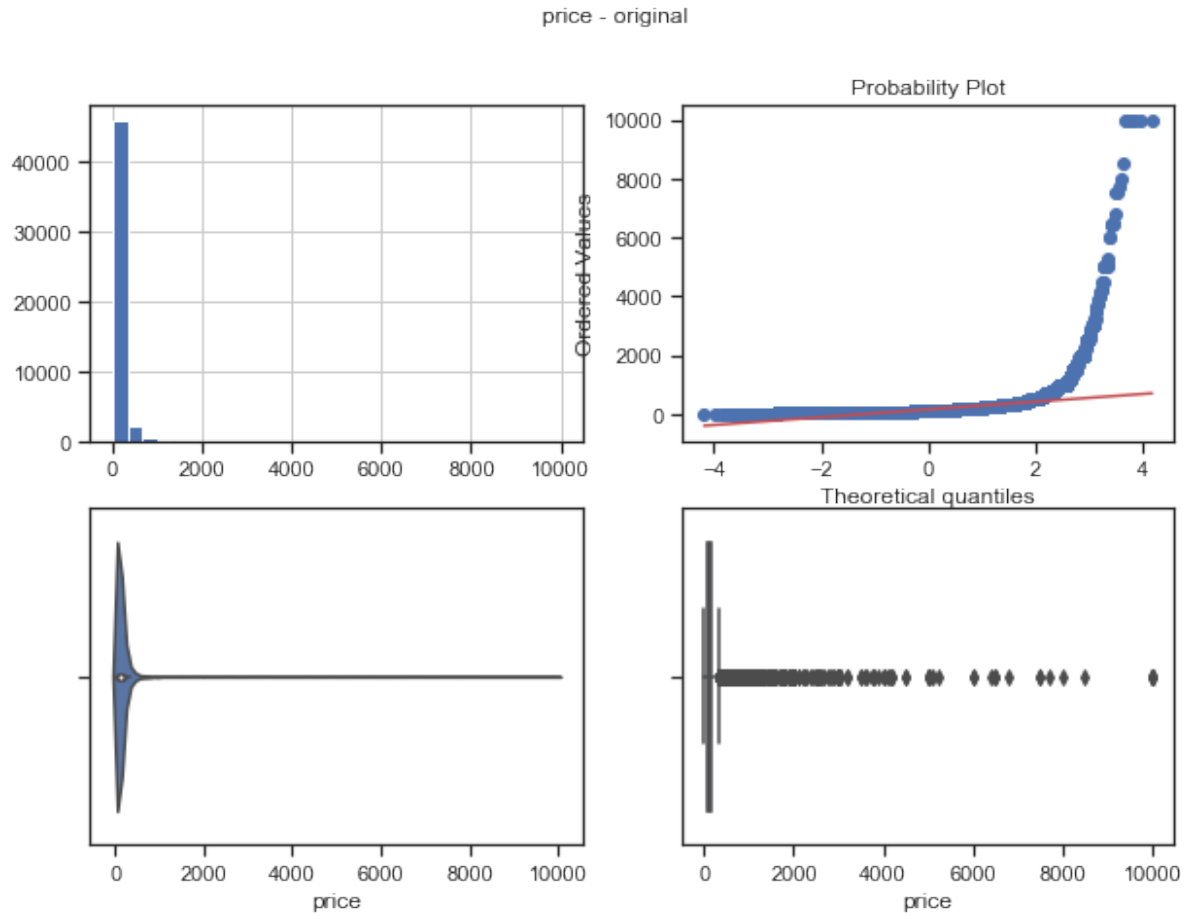
In [23]:
```
data2.describe()
```

Out[23]:

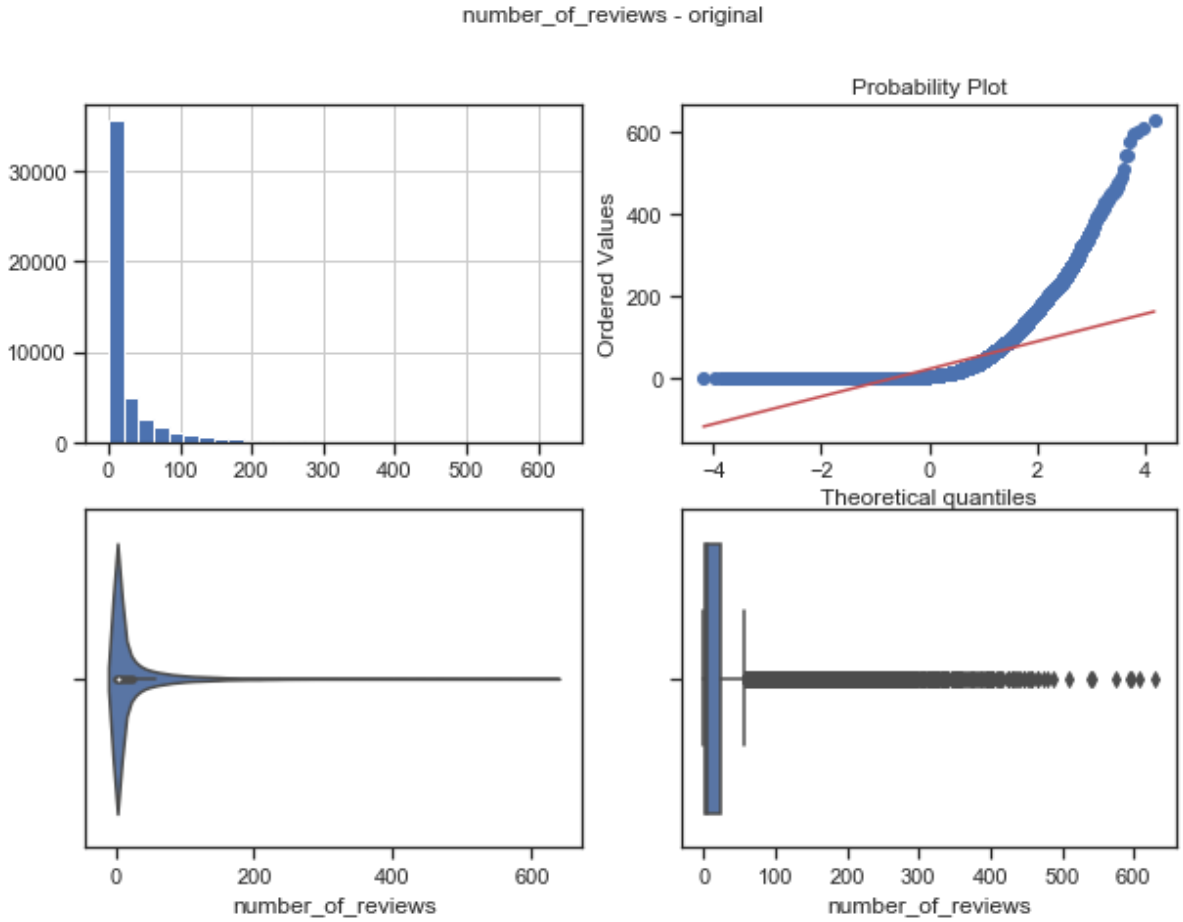
host_id	latitude	longitude	price	minimum_nights	number_of_reviews
.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000
.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466
.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582
.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000
.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000
.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000
.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000
.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000

In [35]:
```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # ящик с усами
    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])
    # ящик с усами
    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

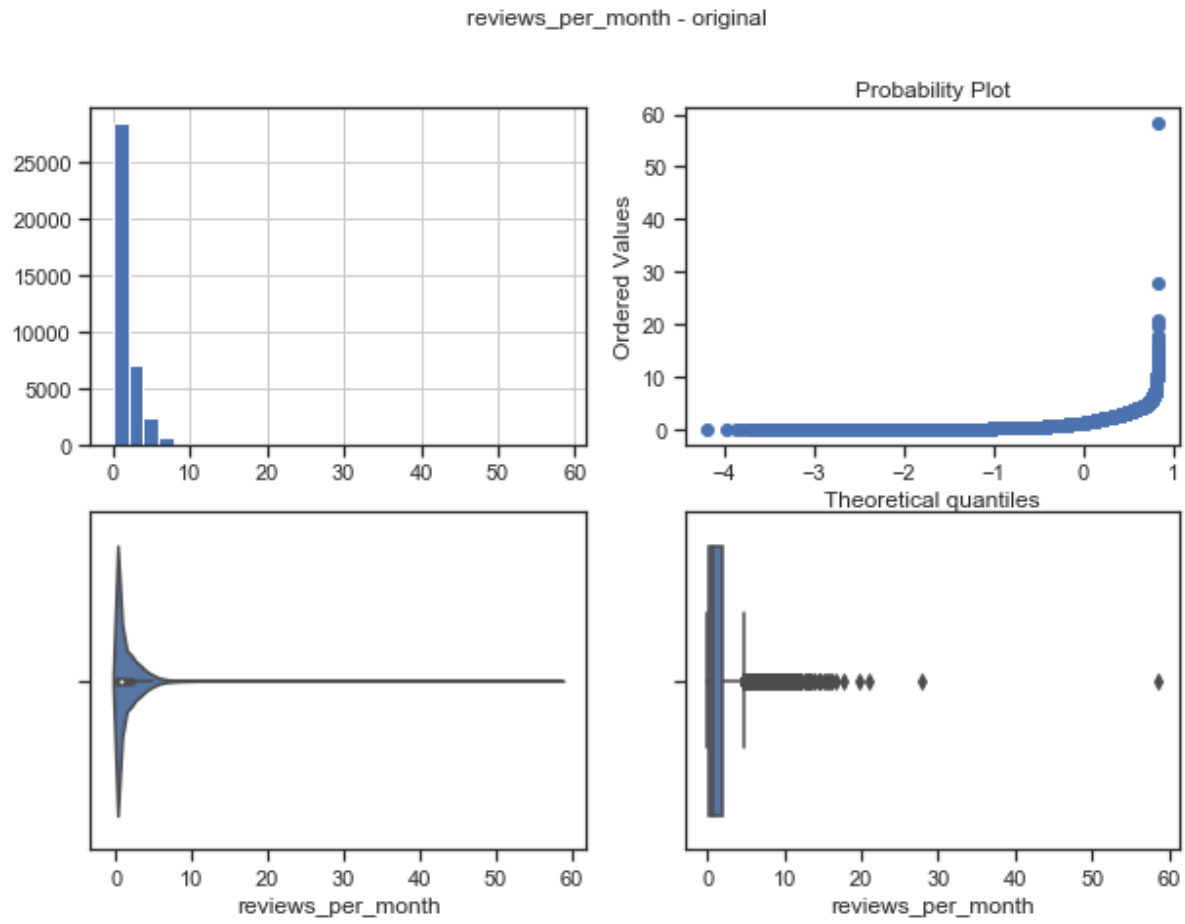
In [39]:
```
diagnostic_plots(data2, 'price', 'price - original')
```



In [40]:
```
diagnostic_plots(data2, 'number_of_reviews', 'number_of_reviews - o
```



In [41]:
```
diagnostic_plots(data2, 'reviews_per_month', 'reviews_per_month - o
```



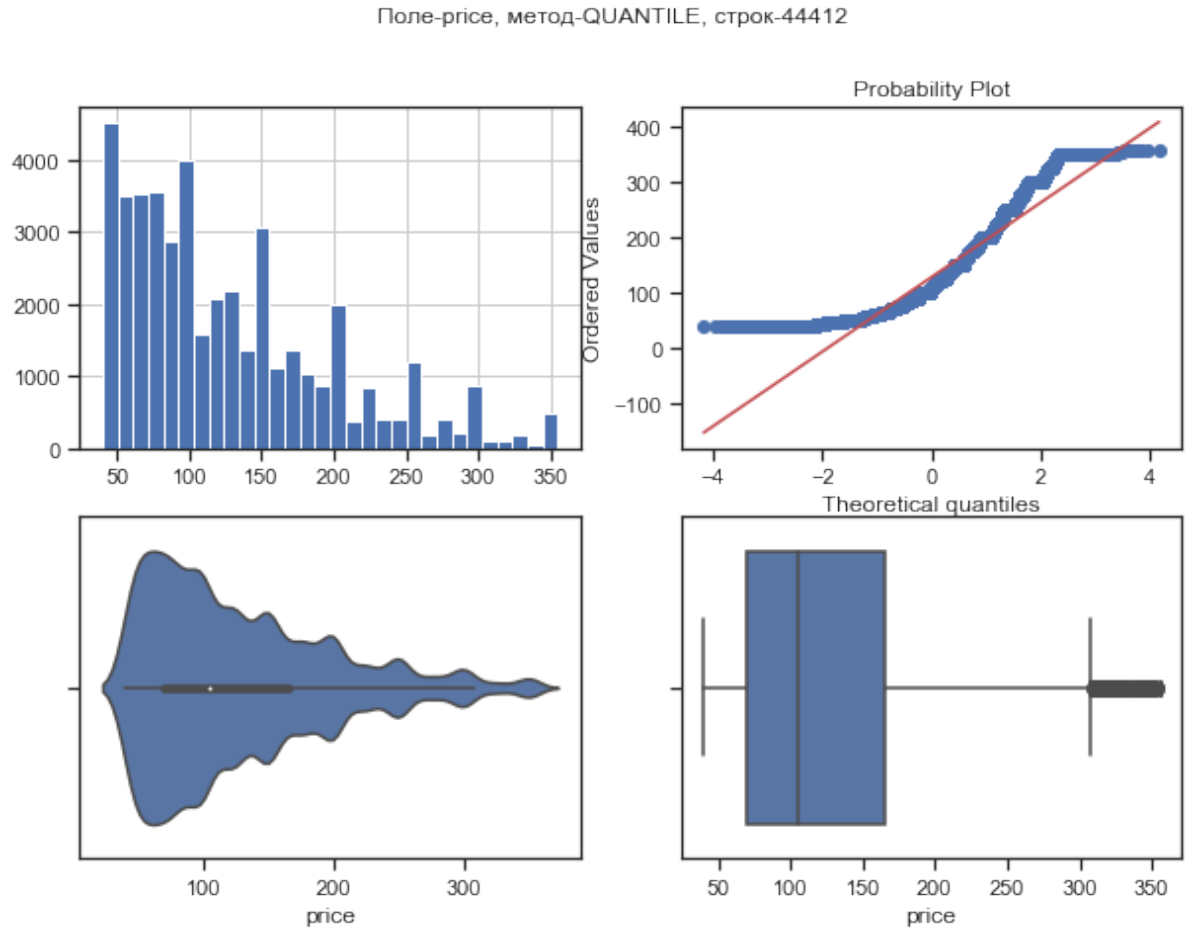
Явно заметны выбросы на полях: number_of_reviews, reviews_per_month, price

In [24]:
```
# Тип вычисления верхней и нижней границы выбросов
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
    QUANTILE = 2
    IRQ = 3
```

In [26]:
```
# Функция вычисления верхней и нижней границы выбросов
def get_outlier_boundaries(df, col):
    lower_boundary = df[col].quantile(0.05)
    upper_boundary = df[col].quantile(0.95)
    return lower_boundary, upper_boundary
```

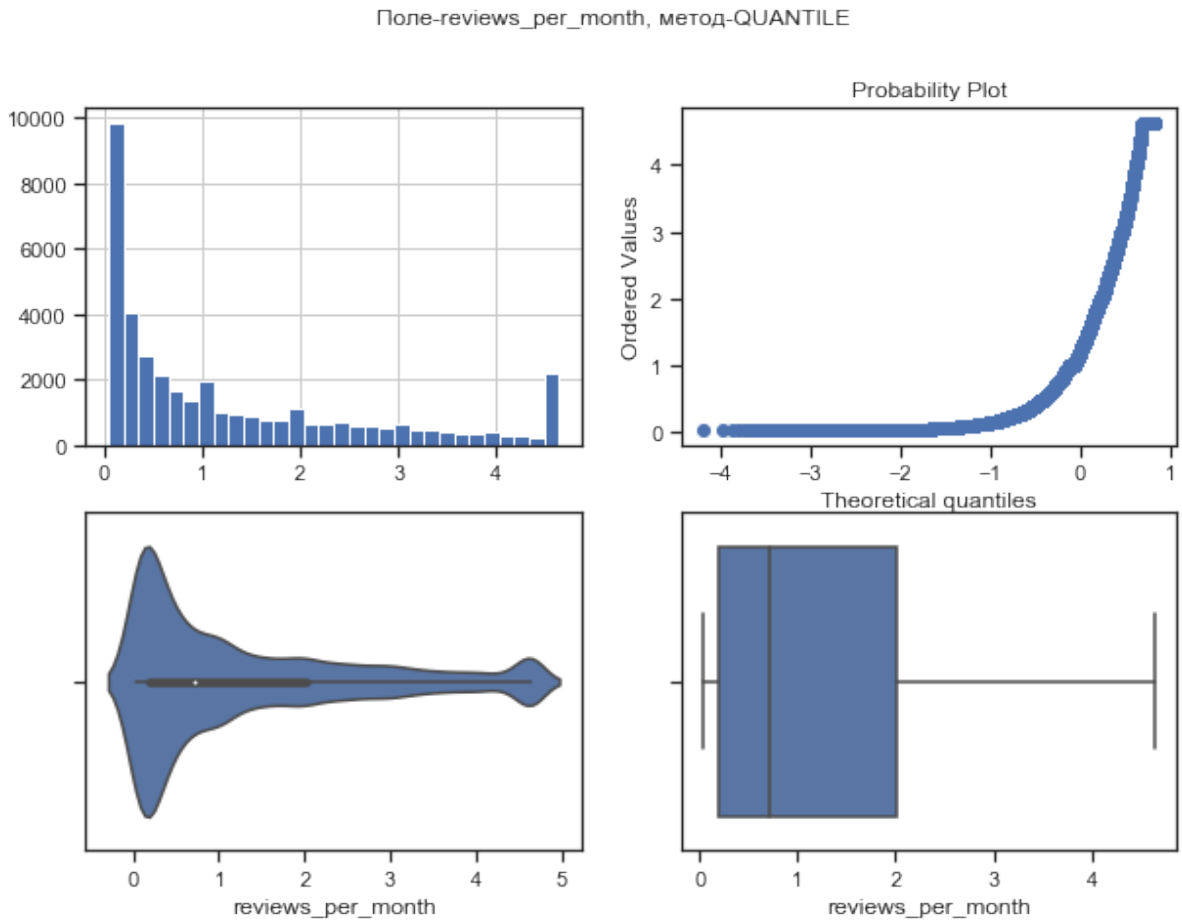
Удаление выбросов (number_of_reviews)

In [38]:
```
# Вычисление верхней и нижней границы
lower_boundary, upper_boundary = get_outlier_boundaries(data2, "pri
# Флаги для удаления выбросов
outliers_temp = np.where(data2["price"] > upper_boundary, True,
                         np.where(data2["price"] < lower_boundary,
# Удаление данных на основе флага
data_trimmed = data2.loc[~(outliers_temp), ]
title = 'Поле-{}, метод-{}, строк-{}'.format("price", "QUANTILE", d
diagnostic_plots(data_trimmed, "price", title)
```



Замена выбросов

In [43]:
```
# Вычисление верхней и нижней границы
lower_boundary, upper_boundary = get_outlier_boundaries(data2, "rev
# Изменение данных
data2["reviews_per_month"] = np.where(data2["reviews_per_month"] >
                        np.where(data2["reviews_per_month"] < lower_bo
title = 'Поле-{}, метод-{}'.format("reviews_per_month", "QUANTILE")
diagnostic_plots(data2, "reviews_per_month", title)
```



Обработка нестандартного признака

In [45]:
```
data2.dtypes
```

Out[45]:
```
id                                  int64
name                               object
host_id                             int64
host_name                          object
neighbourhood_group                object
neighbourhood                      object
latitude                          float64
longitude                         float64
room_type                          object
price                               int64
minimum_nights                      int64
number_of_reviews                   int64
last_review                        object
reviews_per_month                 float64
calculated_host_listings_count      int64
availability_365                    int64
dtype: object
```

In [52]:
```
# Сконвертируем дату и время в нужный формат
data2["last_review_date"] = data2.apply(lambda x: pd.to_datetime(x[
```

In [53]:
```
data2.head(5)
```

Out[53]:

ngitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_mon
3.97237	Private room	149	1	9	2018-10-19	0.2
3.98377	Entire home/apt	225	1	45	2019-05-21	0.3
3.94190	Private room	150	3	0	NaN	Na
3.95976	Entire home/apt	89	1	270	2019-07-05	4.(
3.94399	Entire home/apt	80	10	9	2018-11-19	0.1

In [54]:
```
data2.dtypes
```

Out[54]:
```
id                                 int64
name                              object
host_id                            int64
host_name                         object
neighbourhood_group               object
neighbourhood                     object
latitude                         float64
longitude                        float64
room_type                         object
price                              int64
minimum_nights                     int64
number_of_reviews                  int64
last_review                       object
reviews_per_month                float64
calculated_host_listings_count     int64
availability_365                   int64
last_review_date           datetime64[ns]
dtype: object
```

In [55]:
```
# День
data2['last_review_day'] = data2['last_review_date'].dt.day
# Месяц
data2['last_review_month'] = data2['last_review_date'].dt.month
# Год
data2['last_review_year'] = data2['last_review_date'].dt.year
```

In [56]:
```
data2.head(5)
```

Out[56]:

›f_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365	last
9	2018-10-19	0.21	6	365	
45	2019-05-21	0.38	2	355	
0	NaN	NaN	1	365	
270	2019-07-05	4.64	1	194	
9	2018-11-19	0.10	1	0	