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)»

по дисциплине

«Методы машинного обучения»

ИСПОЛНИТЕЛЬ:

Крюков Г.М.
Группа ИУ5-21М

____

"__"___________2022 г.

Москва, 2022

<u>Цель:</u> Изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

<u>Задание:</u>

1. Выбрать один или несколько наборов данных (датасетов) для решения следующих задач. Каждая задача может быть решена на отдельном датасете, или несколько задач могут быть решены на одном датасете. Просьба не использовать датасет, на котором данная задача решалась в лекции.
2. 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).

<u>Набор данных:</u>

This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient.

<u>Поля</u>:
id: unique identifier
gender: "Male", "Female" or "Other"
age: age of the patient
hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart
disease
ever_married: "No" or "Yes"
work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
Residence_type: "Rural" or "Urban"
avg_glucose_level: average glucose level in blood
bmi: body mass index
smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
stroke: 1 if the patient had a stroke or 0 if not
"Unknown" in smoking_status means that the information is unavailable for this patient

Текст программы:

```
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
```

```
[ ] !pip install numpy==1.16.4
```

```
Collecting numpy==1.16.4
  Downloading numpy-1.16.4-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (13.9 MB)
     |████████████████████████████████| 13.9 MB 3.5 MB/s eta 0:00:01
ERROR: statsmodels 0.12.2 has requirement scipy>=1.1, but you'll have scipy 1.0.0 which is incompatible.
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.19.5
    Uninstalling numpy-1.19.5:
      Successfully uninstalled numpy-1.19.5
Successfully installed numpy-1.16.4
```

```
[ ] data = pd.read_csv('/Users/user/Downloads/data_stroke.csv')
```

```
[ ] data.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	NaN	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	NaN	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

```
[ ] data = data.drop('id', 1)
    data.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	NaN	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	Female	NaN	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

```
[ ] # Заполним пропуски
    data.dropna(subset=['age'], inplace=True)
```

```
[ ] data['gender'] = data['gender'].astype(str).str[0]
```

```
[ ] # Заполним пропуски возраста средними значениями
    def impute_na(df, variable, value):
        df[variable].fillna(value, inplace=True)
    impute_na(data, 'bmi', data['bmi'].mean())
```

```
[ ] data.describe()
```

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5094.000000	5094.000000	5094.000000	5094.000000	5094.000000	5094.000000
mean	43.182960	0.097173	0.053592	106.074751	28.886269	0.046918
std	22.601491	0.296222	0.225234	45.216297	7.697727	0.211484
min	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	25.000000	0.000000	0.000000	77.265000	23.800000	0.000000
50%	45.000000	0.000000	0.000000	91.850000	28.400000	0.000000
75%	61.000000	0.000000	0.000000	114.017500	32.800000	0.000000
max	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

```
[ ] X_ALL = data.drop(['stroke', 'gender', 'ever_married', 'work_type','Residence_type','smoking_status'], axis=1)
```

```
[ ] # Функция для восстановления датафрейма
    # на основе масштабированных данных
    def arr_to_df(arr_scaled):
        res = pd.DataFrame(arr_scaled, columns=X_ALL.columns)
        return res
```

```
[ ] # Разделим выборку на обучающую и тестовую
    X_train, X_test, y_train, y_test = train_test_split(X_ALL, data['stroke'],
                                                        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
```

```
((4075, 5), (1019, 5))
```

StandardScaler

```
[ ] # 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
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/data.py:645: DataConversi
  return self.partial_fit(X, y)
/anaconda3/lib/python3.6/site-packages/sklearn/base.py:464: DataConversionWarning: Dat
  return self.fit(X, **fit_params).transform(X)
```

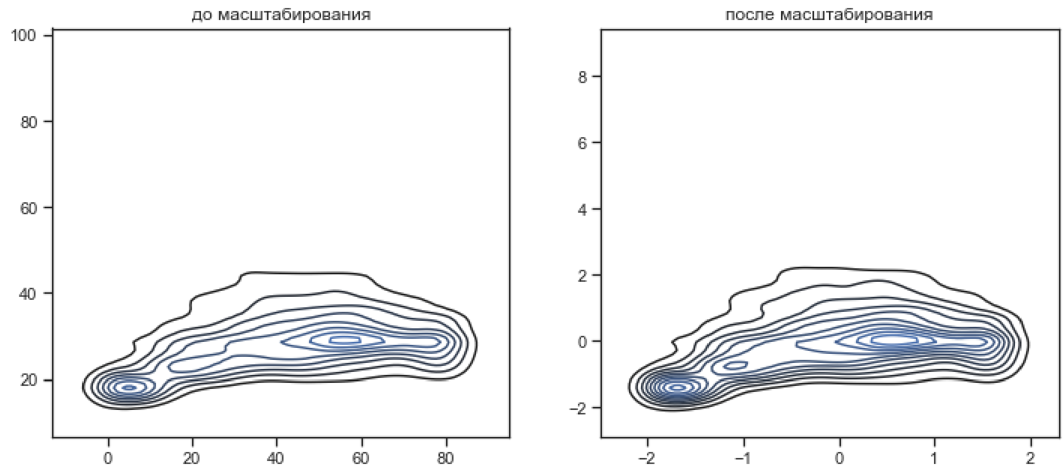
	age	hypertension	heart_disease	avg_glucose_level	bmi
0	0.788390	-0.328073	-0.237965	2.126328	3.231011e-15
1	1.629125	-0.328073	4.202302	-0.003423	4.695004e-01
2	0.257399	-0.328073	-0.237965	1.441110	7.163508e-01
3	1.673374	-0.328073	-0.237965	1.772439	1.477609e-02
4	1.363630	3.048099	4.202302	-0.795914	-1.930979e-01
5	1.142384	-0.328073	-0.237965	-0.258444	-7.907356e-01
6	0.699892	-0.328073	-0.237965	-0.661878	3.231011e-15
7	1.540627	-0.328073	-0.237965	-1.050714	-6.088458e-01
8	1.673374	3.048099	-0.237965	-0.567213	1.057210e-01
9	0.788390	-0.328073	4.202302	0.318174	1.028162e+00
10	0.478645	-0.328073	-0.237965	-0.034609	-2.060900e-01
11	1.584876	-0.328073	4.202302	2.389091	-8.916090e-02
12	0.301649	3.048099	-0.237965	1.356619	2.616264e-01
5086	0.611393	-0.328073	-0.237965	-0.622508	-9.336489e-01
5087	-1.114326	-0.328073	-0.237965	-0.513687	2.340366e+00
5088	-1.335572	-0.328073	-0.237965	-0.066238	-1.336405e+00
5089	1.629125	3.048099	-0.237965	-0.493781	3.231011e-15
5090	1.673374	-0.328073	-0.237965	0.423014	1.443910e+00
5091	-0.362090	-0.328073	-0.237965	-0.510591	2.226501e-01
5092	0.345898	-0.328073	-0.237965	1.331846	-4.269561e-01
5093	0.036153	-0.328073	-0.237965	-0.459940	-3.490034e-01

5094 rows × 5 columns

```
[ ] # 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()
```

```
[ ] draw_kde(['age', 'bmi'], data, data_cs11_scaled, 'до масштабирования', 'после масштабирования')
```

```
/anaconda3/lib/python3.6/site-packages/seaborn/distributions.py:679: UserWarning: Passing a 2D dataset
  warnings.warn(warn_msg, UserWarning)
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarning: Using a non-tuple sec
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



Масштабирование "Mean Normalisation"

```
[ ] # Разделим выборку на обучающую и тестовую
    X_train, X_test, y_train, y_test = train_test_split(X_ALL, data['stroke'],
                                                        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
```

```
((4075, 5), (1019, 5))
```

```
[ ] 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)
```

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

	age	hypertension	heart_disease	avg_glucose_level	bmi
count	5094.000000	5094.000000	5094.000000	5094.000000	5094.000000
mean	0.000239	0.003431	0.001323	0.001318	0.000032
std	0.275897	0.296222	0.225234	0.208736	0.088176
min	-0.525921	-0.093742	-0.052270	-0.233909	-0.212869
25%	-0.221721	-0.093742	-0.052270	-0.131679	-0.058230
50%	0.022419	-0.093742	-0.052270	-0.064349	-0.005538
75%	0.217732	-0.093742	-0.052270	0.037985	0.044863
max	0.474079	0.906258	0.947730	0.766091	0.787131

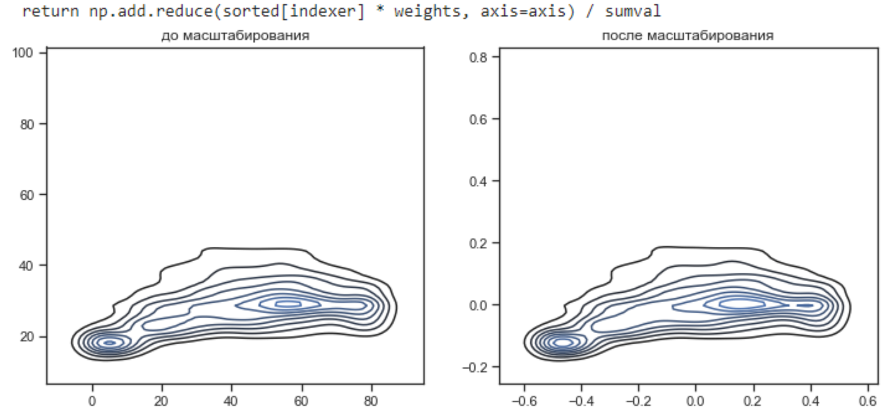
```
[ ] cs22 = MeanNormalisation()
    cs22.fit(X_train)
    data_cs22_scaled_train = cs22.transform(X_train)
    data_cs22_scaled_test = cs22.transform(X_test)
```

```
[ ] data_cs22_scaled_train.describe()
```

	age	hypertension	heart_disease	avg_glucose_level	bmi
count	4.075000e+03	4.075000e+03	4.075000e+03	4.075000e+03	4.075000e+03
mean	-2.867645e-16	2.205257e-16	2.604937e-17	1.395040e-15	4.274188e-16
std	2.747227e-01	2.915057e-01	2.225982e-01	2.089053e-01	8.717950e-02
min	-5.259206e-01	-9.374233e-02	-5.226994e-02	-2.339087e-01	-2.128694e-01
25%	-2.217214e-01	-9.374233e-02	-5.226994e-02	-1.329715e-01	-5.823027e-02
50%	2.241924e-02	-9.374233e-02	-5.226994e-02	-6.527235e-02	-5.538399e-03
75%	2.177317e-01	-9.374233e-02	-5.226994e-02	3.541088e-02	4.486252e-02
max	4.740794e-01	9.062577e-01	9.477301e-01	7.660913e-01	7.871306e-01

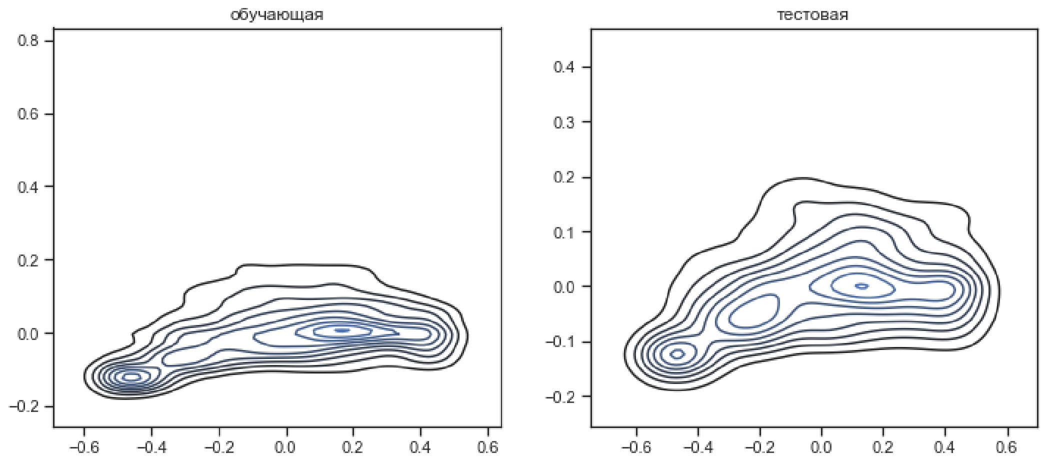
```
[ ] draw_kde(['age', 'bmi'], data, data_cs21_scaled, 'до масштабирования', 'после масштабирования')

/anaconda3/lib/python3.6/site-packages/seaborn/distributions.py:679: UserWarning: Passing a 2D data:
  warnings.warn(warn_msg, UserWarning)
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarning: Using a non-tuple :
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



```
[ ] draw_kde(['age', 'bmi'], data_cs22_scaled_train, data_cs22_scaled_test, 'обучающая', 'тестовая')
```

```
/anaconda3/lib/python3.6/site-packages/seaborn/distributions.py:679: UserWarning: Passing a 2D dataset f
  warnings.warn(warn_msg, UserWarning)
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarning: Using a non-tuple seque
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



MinMax-масштабирование

```
[ ] # 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()
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/data.py:334: DataConversionWar
  return self.partial_fit(X, y)
```

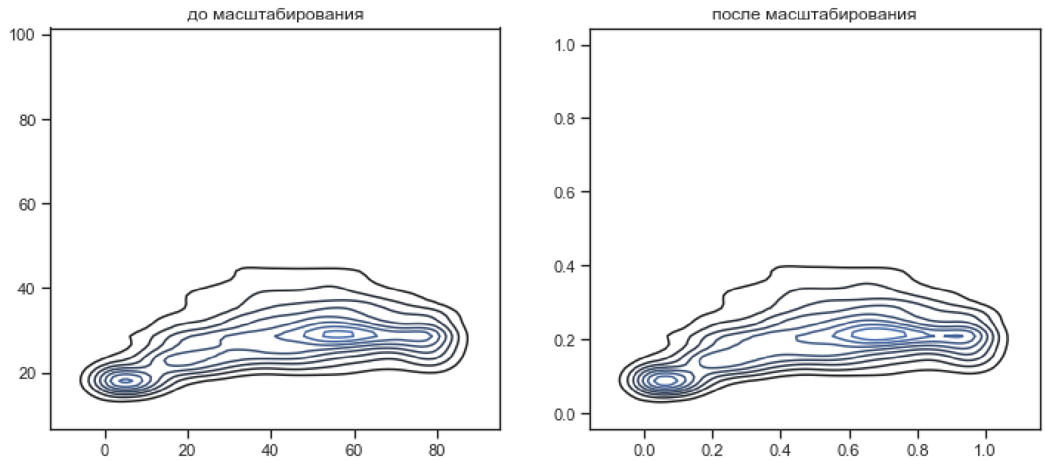
	age	hypertension	heart_disease	avg_glucose_level	bmi
count	5094.000000	5094.000000	5094.000000	5094.000000	5094.000000
mean	0.526159	0.097173	0.053592	0.235226	0.212901
std	0.275897	0.296222	0.225234	0.208736	0.088176
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.304199	0.000000	0.000000	0.102230	0.154639
50%	0.548340	0.000000	0.000000	0.169560	0.207331
75%	0.743652	0.000000	0.000000	0.271893	0.257732
max	1.000000	1.000000	1.000000	1.000000	1.000000

```
[ ] 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)
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/data.py:334: DataConversionWarning: Data wi
  return self.partial_fit(X, y)
```

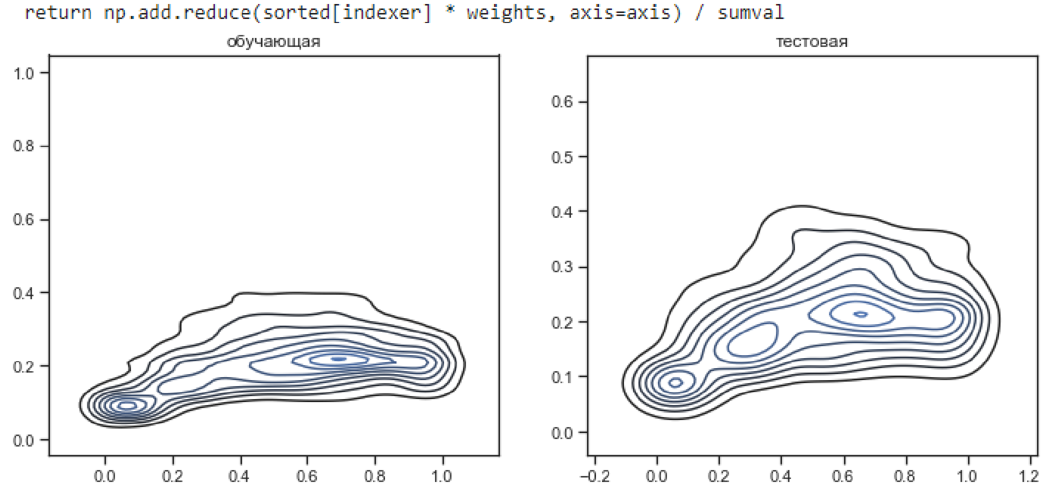
```
draw_kde(['age', 'bmi'], data, data_cs31_scaled, 'до масштабирования', 'после масштабирования')
```

```
/anaconda3/lib/python3.6/site-packages/seaborn/distributions.py:679: UserWarning: Passing a 2D dataset f
  warnings.warn(warn_msg, UserWarning)
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarning: Using a non-tuple seque
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```




```
[ ] draw_kde(['age', 'bmi'], data_cs32_scaled_train, data_cs32_scaled_test, 'обучающая', 'тестовая')
```

```
/anaconda3/lib/python3.6/site-packages/seaborn/distributions.py:679: UserWarning: Passing a 2D dataset
  warnings.warn(warn_msg, UserWarning)
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarning: Using a non-tuple sequ
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



Обработка выбросов для числовых признаков

```
[ ] data2 = pd.read_csv('/Users/user/Downloads/AB_NYC_2019.csv')
```

```
[ ] data2.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	n
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	

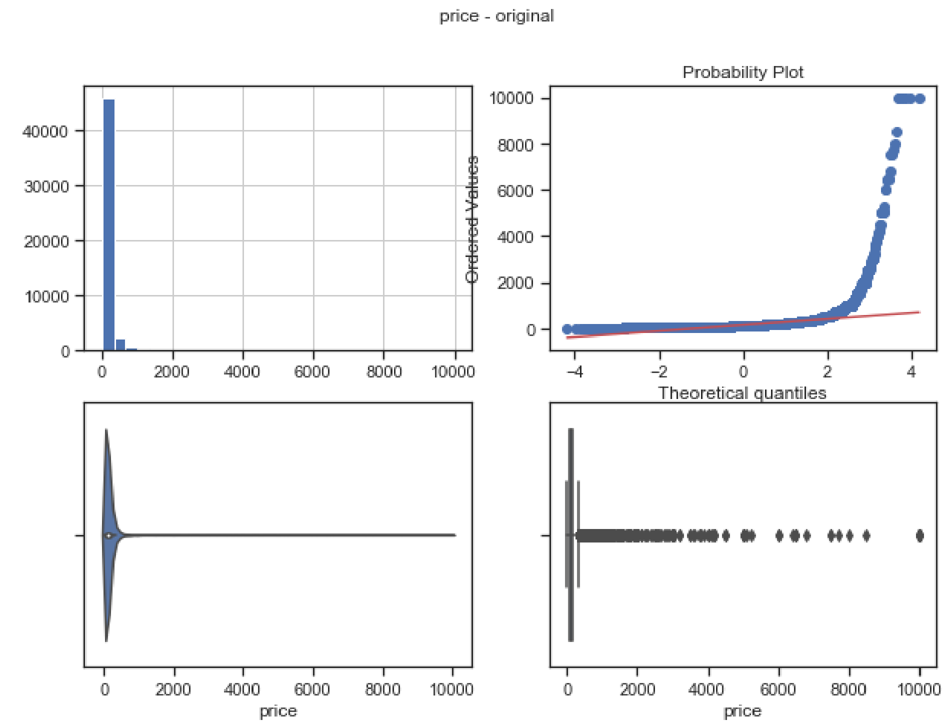
```
[ ] data2.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000	48895.000000
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	7.143982	112.781327
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	32.952519	131.622289
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	1.000000	0.000000
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	1.000000	0.000000
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	1.000000	45.000000
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	2.000000	227.000000
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	327.000000	365.000000

```python
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()
```

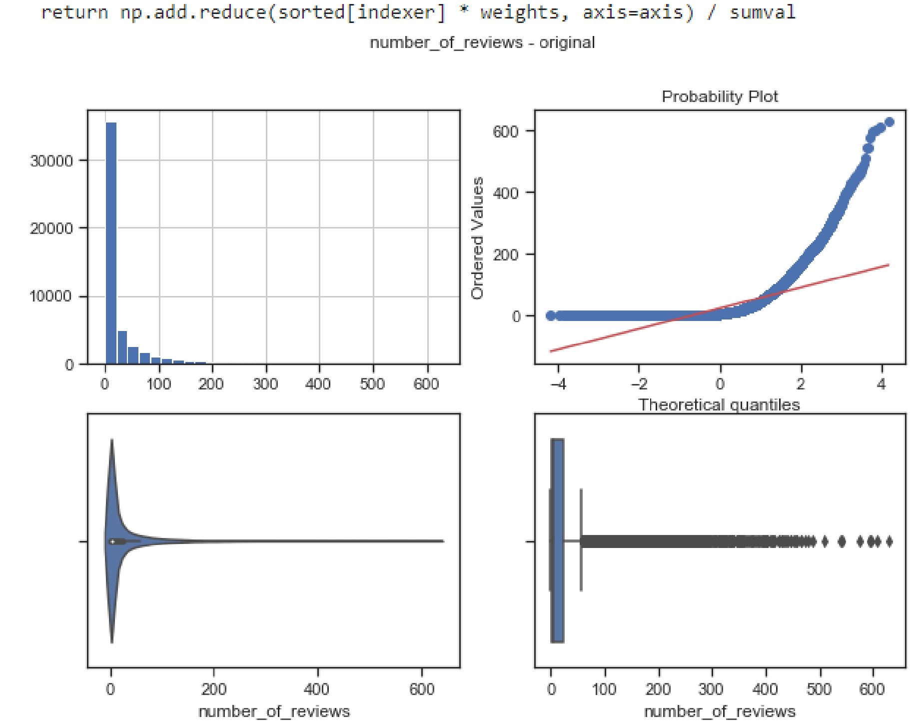
```
[ ]  diagnostic_plots(data2, 'price', 'price - original')
```

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarning: Using a non-tu
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



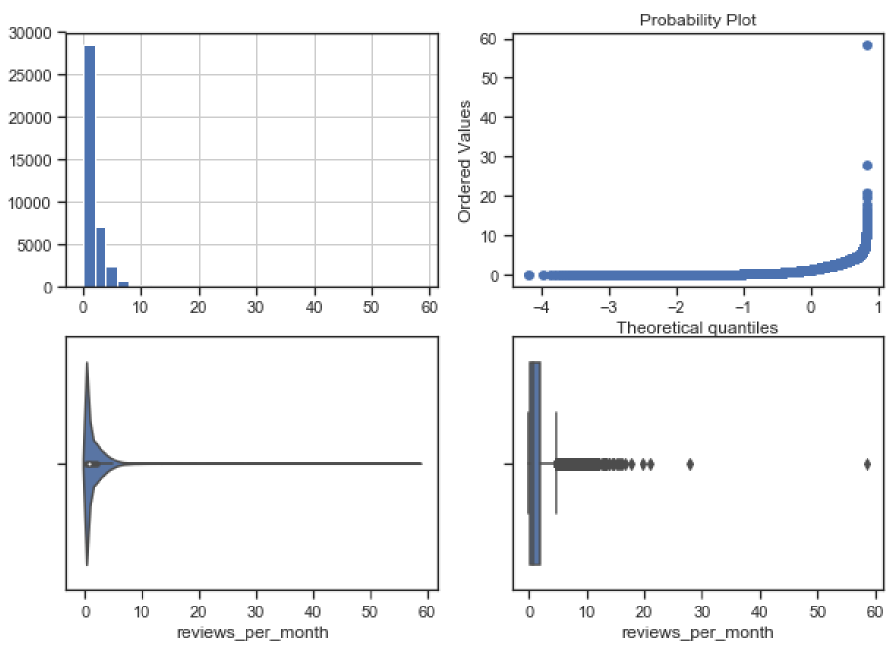

```
diagnostic_plots(data2, 'number_of_reviews', 'number_of_reviews - original')
```

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarning: Using a nor
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



```
[ ] diagnostic_plots(data2, 'reviews_per_month', 'reviews_per_month - original')
```

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarning: Using a non-
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

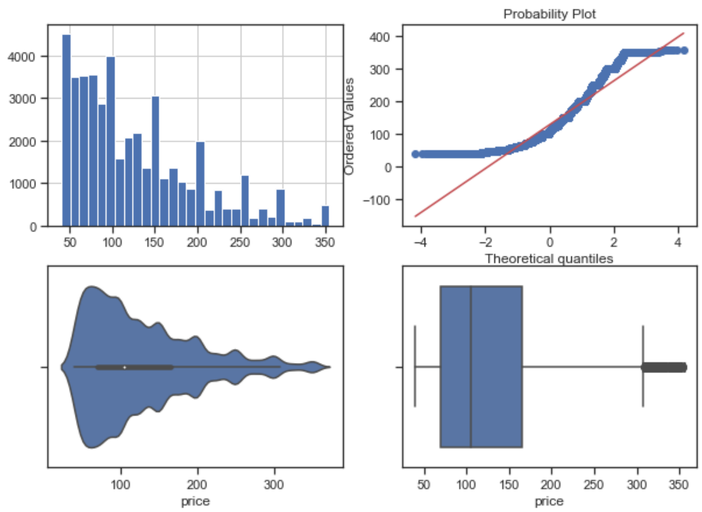



```
[ ] # Тип вычисления верхней и нижней границы выбросов
    from enum import Enum
    class OutlierBoundaryType(Enum):
        SIGMA = 1
        QUANTILE = 2
        IRQ = 3
```

```
[ ] # 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)

```
[ ] # Вычисление верхней и нижней границы
    lower_boundary, upper_boundary = get_outlier_boundaries(data2, "price")
    # Флаги для удаления выбросов
    outliers_temp = np.where(data2["price"] > upper_boundary, True,
                             np.where(data2["price"] < lower_boundary, True, False))
    # Удаление данных на основе флага
    data_trimmed = data2.loc[~(outliers_temp), ]
    title = 'Поле-{}, метод-{}, строк-{}'.format("price", "QUANTILE", data_trimmed.shape[0])
    diagnostic_plots(data_trimmed, "price", title)
```

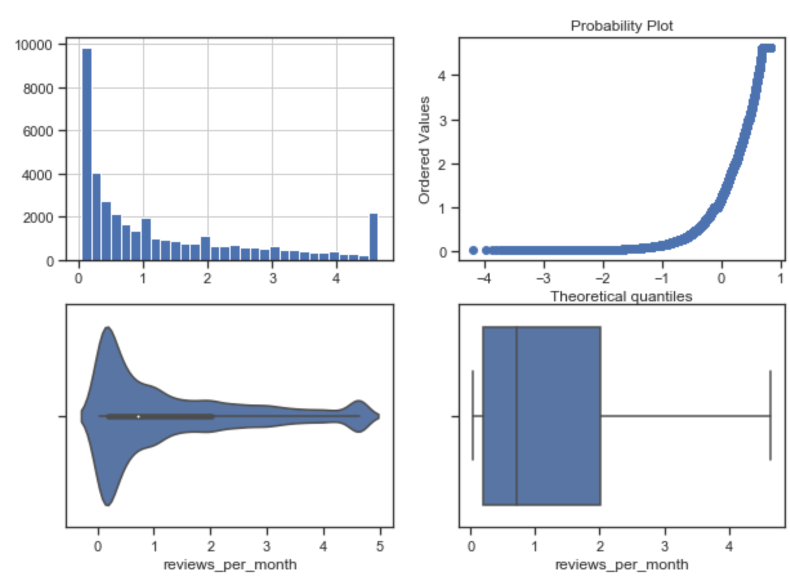


Замена выбросов

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

```
/anaconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1706: FutureWarning: Using a non-tuple sequence for multidi
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Поле-reviews_per_month, метод-QUANTILE



Обработка нестандартного признака

```
[ ] data2.dtypes
```

```
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
```

```
[ ] # Сконвертируем дату и время в нужный формат
    data2["last_review_date"] = data2.apply(lambda x: pd.to_datetime(x["last_review"], format='%Y/%m/%d'), axis=1)
```

```
[ ] data2.head(5)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	numbe
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	

```
[ ] data2.dtypes
```

```
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
```

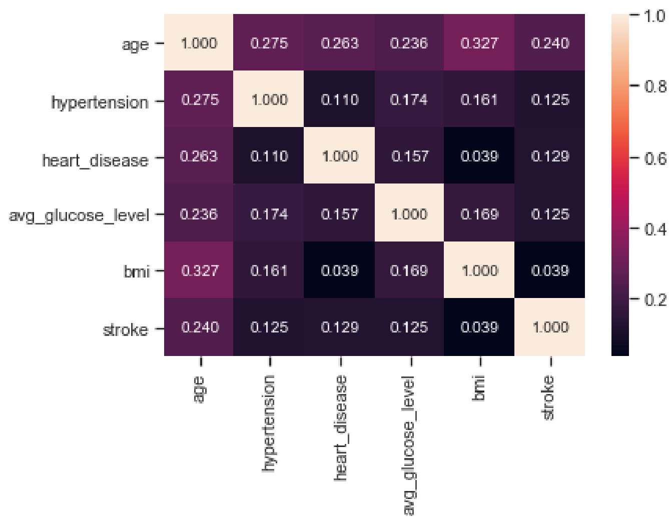
```
[ ] # День
    data2['last_review_day'] = data2['last_review_date'].dt.day
    # Месяц
    data2['last_review_month'] = data2['last_review_date'].dt.month
    # Год
    data2['last_review_year'] = data2['last_review_date'].dt.year
```

Отбор признаков

Метод фильтрации (Корреляция признаков)

```
[ ] sns.heatmap(data.corr(), annot=True, fmt='.3f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a27e15278>
```




```
[ ] # Формирование DataFrame с сильными корреляциями
    def make_corr_df(df):
        cr = data.corr()
        cr = cr.abs().unstack()
        cr = cr.sort_values(ascending=False)
        cr = cr[cr >= 0.3]
        cr = cr[cr < 1]
        cr = pd.DataFrame(cr).reset_index()
        cr.columns = ['f1', 'f2', 'corr']
        return cr
```

```
[ ] # Обнаружение групп коррелирующих признаков
    def corr_groups(cr):
        grouped_feature_list = []
        correlated_groups = []

        for feature in cr['f1'].unique():
            if feature not in grouped_feature_list:
                # находим коррелирующие признаки
                correlated_block = cr[cr['f1'] == feature]
                cur_dups = list(correlated_block['f2'].unique()) + [feature]
                grouped_feature_list = grouped_feature_list + cur_dups
                correlated_groups.append(cur_dups)
        return correlated_groups
```

```
[ ] # Группы коррелирующих признаков
    corr_groups(make_corr_df(data))
```

```
[['age', 'bmi']]
```

Метод из группы методов вложений

```
[ ] # Используем L1-регуляризацию
    e_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1', max_iter=500, random_state=1)
    e_lr1.fit(X_train, y_train)
    # Коэффициенты регрессии
    e_lr1.coef_

    array([[0.06566053, 0.43919476, 0.37663459, 0.00437325, 0.01514052]])
```

```
[ ] # Все 4 признака являются "хорошими"
    from sklearn.feature_selection import SelectFromModel
    sel_e_lr1 = SelectFromModel(e_lr1)
    sel_e_lr1.fit(X_train, y_train)
    sel_e_lr1.get_support()

    array([ True,  True,  True,  True,  True])
```

Вывод:

При выполнении работы выбран датасет для предсказания вероятности проявления у человека сердечного приступа и выполнены следующие задачи:

- масштабирование признаков (StandardScaler, Mean Normalisation, MinMax)
- обработка выбросов для числовых признаков (удаление и замена)
- обработка нестандартного признака
- отбор признаков