Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)"
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА ___СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)_____

Отчет по лабораторной работе №3

«Обработка признаков (Часть 2)»

по курсу «Методы машинного обучения».

ИСПОЛНИТЕЛЬ: Группа ИУ5-24М
Уралова Е.А.
ФИО
_____
подпись

"12" марта 2024 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.Е.
ФИО
_____
подпись

"__"_____2024 г.

Москва – 2024

Задание:

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.

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).

Выполнение:

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
```

In [2]:
```
!pip install numpy==1.16.4
```

```
Collecting numpy==1.16.4
  Using cached numpy-1.16.4.zip (5.1 MB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: numpy
  Building wheel for numpy (setup.py) ... error
  error: subprocess-exited-with-error

  × python setup.py bdist_wheel did not run successfully.
  │ exit code: 1
  └─> [2223 lines of output]
      Running from numpy source directory.
      /private/var/folders/z0/w0jmy3853hl1_sp6r8x9hswm0000gn/T/pip-install-nn1ovv55/numpy_ae2b4b0a0ea449bcaa6
2beb53fd6c5c2/numpy/distutils/misc_util.py:476: SyntaxWarning: "is" with a literal. Did you mean "=="?
        return is_string(s) and ('*' in s or '?' is s)
      blas_opt_info:
      blas_mkl_info:
      customize UnixCCompiler
        FOUND:
          libraries = ['mkl_rt', 'pthread']
          library_dirs = ['/Users/irinaelkhimova/opt/anaconda3/lib']
          define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
```

In [3]:
```
data = pd.read_csv('heart2.csv')
```

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

Out[4]:

	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

In [5]:
```
data = data.drop('id', 1)
data.head()
```

```
/var/folders/z0/w0jmy3853hl1_sp6r8x9hswm0000gn/T/ipykernel_4636/3892771371.py:1: FutureWarning: In a future v
ersion of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  data = data.drop('id', 1)
```

Out[5]:

	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

In [6]:
```
# Заполним пропуски
data.dropna(subset=['age'], inplace=True)
```

In [7]:
```
data['gender'] = data['gender'].astype(str).str[0]
```

In [8]:
```
# Заполним пропуски возраста средними значениями
def impute_na(df, variable, value):
    df[variable].fillna(value, inplace=True)
impute_na(data, 'bmi', data['bmi'].mean())
```

In [9]:
```
data.describe()
```

Out[9]:

	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

In [10]:
```
X_ALL = data.drop(['stroke', 'gender', 'ever_married', 'work_type','Residence_type','smoking_status'], axis=1)
```

In [11]:
```
# 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
```

In [12]:
```
# 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
```

Out[12]:
```
((4075, 5), (1019, 5))
```

StandardScaler

In [13]:
```
# 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[13]:

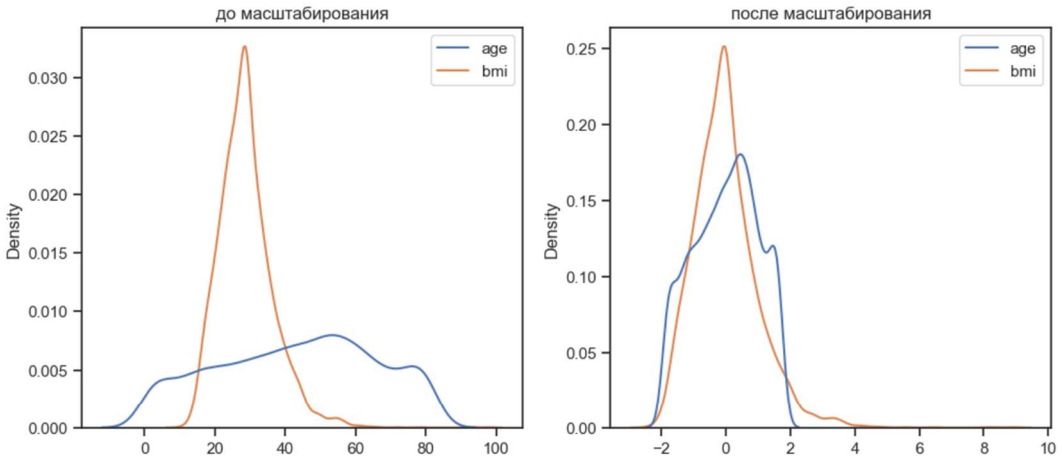
	age	hypertension	heart_disease	avg_glucose_level	bmi
0	0.788390	-0.328073	-0.237965	2.126328	3.231011e-15
1	1.629125	-0.328073	4.202302	-0.003423	4.695004e-01
2	0.257399	-0.328073	-0.237965	1.441110	7.163508e-01
3	1.673374	-0.328073	-0.237965	1.772439	1.477609e-02
4	1.363630	3.048099	4.202302	-0.795914	-1.930979e-01
...	...	...	...	...	...
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

In [14]:
```
# 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)
```

```
    # 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 [15]:
```
draw_kde(['age', 'bmi'], data, data_cs11_scaled, 'до масштабирования', 'после масштабирования')
```



Масштабирование "Mean Normalisation"

In [16]:
```
# 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
```

Out[16]:
```
((4075, 5), (1019, 5))
```

In [17]:
```
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 [18]:
```
sc21 = MeanNormalisation()
data_cs21_scaled = sc21.fit_transform(X_ALL)
data_cs21_scaled.describe()
```

Out[18]:

	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

In [19]:
```
cs22 = MeanNormalisation()
cs22.fit(X_train)
data_cs22_scaled_train = cs22.transform(X_train)
data_cs22_scaled_test = cs22.transform(X_test)
```

In [20]:
```
data_cs22_scaled_train.describe()
```

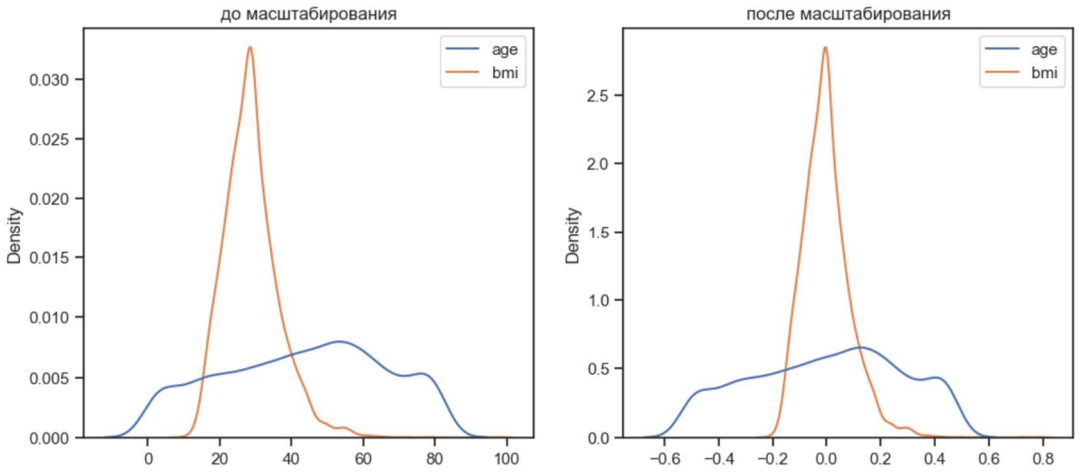
Out[20]:

In [20]:
```
data_cs22_scaled_train.describe()
```

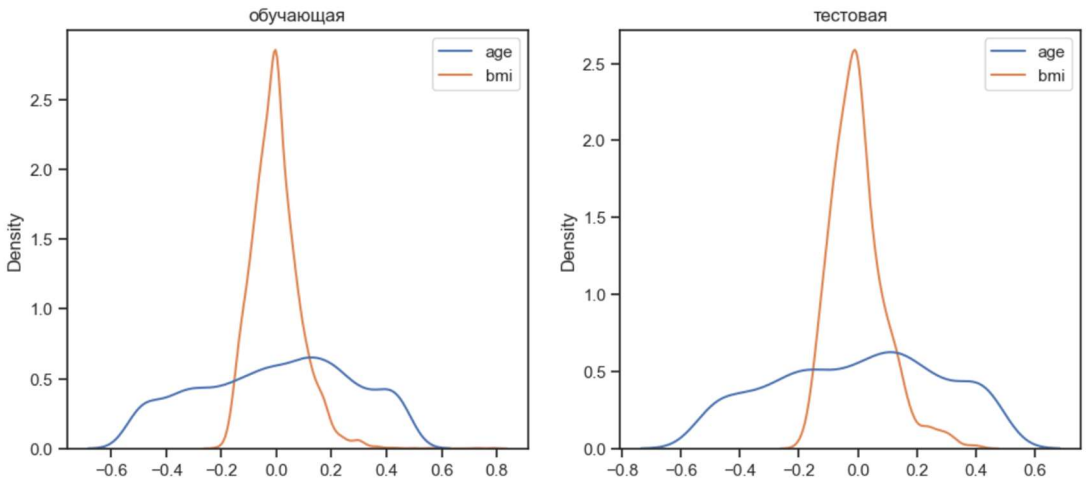
Out[20]:

	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

In [21]:
```
draw_kde(['age', 'bmi'], data, data_cs21_scaled, 'до масштабирования', 'после масштабирования')
```



In [22]:
```
draw_kde(['age', 'bmi'], data_cs22_scaled_train, data_cs22_scaled_test, 'обучающая', 'тестовая')
```



MinMax-масштабирование

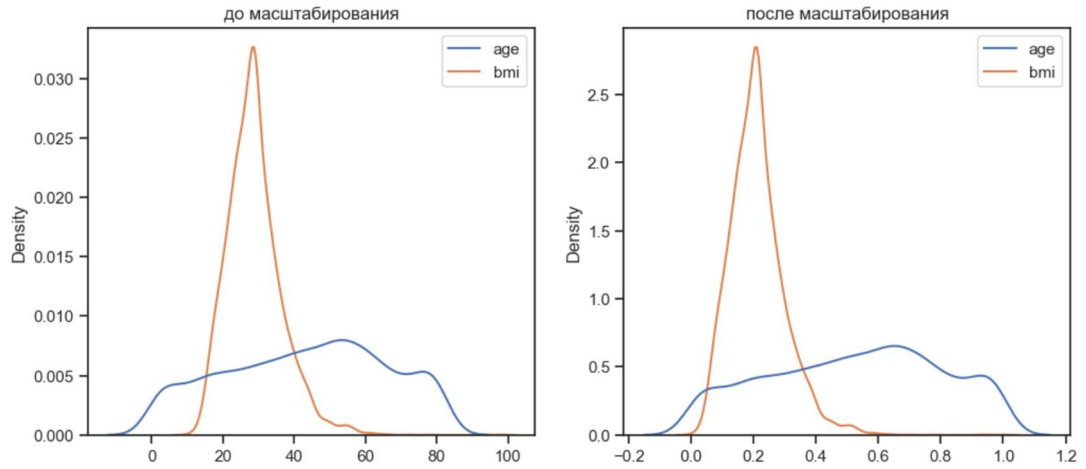
In [23]:
```
# Обучаем StandardScaler на всей выборке и масштабируем
cs31 = MinMaxScaler()
data_cs31_scaled_temp = cs31.fit_transform(X_ALL)
# формируем DataFrame на основе массива
data_cs31_scaled = arr_to_df(data_cs31_scaled_temp)
data_cs31_scaled.describe()
```

Out[23]:

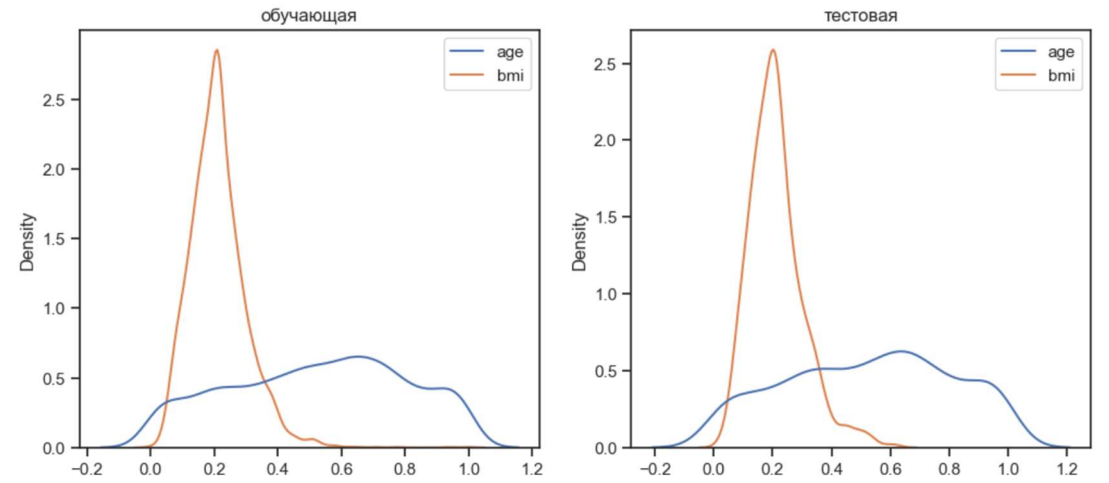
	age	hypertension	heart_disease	avg_glucose_level	bmi
count	5094.000000	5094.000000	5094.000000	5094.000000	5094.000000
mean	0.526159	0.097173	0.053592	0.235226	0.212901
std	0.275897	0.296222	0.225234	0.208736	0.088176
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.304199	0.000000	0.000000	0.102230	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

In [24]:
```
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 [25]:
```
draw_kde(['age', 'bmi'], data, data_cs31_scaled, 'до масштабирования', 'после масштабирования')
```



In [26]:
```
draw_kde(['age', 'bmi'], data_cs32_scaled_train, data_cs32_scaled_test, 'обучающая', 'тестовая')
```



Обработка выбросов для числовых признаков ¶

In [28]:
```
data2 = pd.read_csv('heart.csv')
```

In [29]:
```
data2.head()
```

Out[29]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289.0	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180.0	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283.0	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214.0	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	NaN	0	Normal	122	N	0.0	Up	0

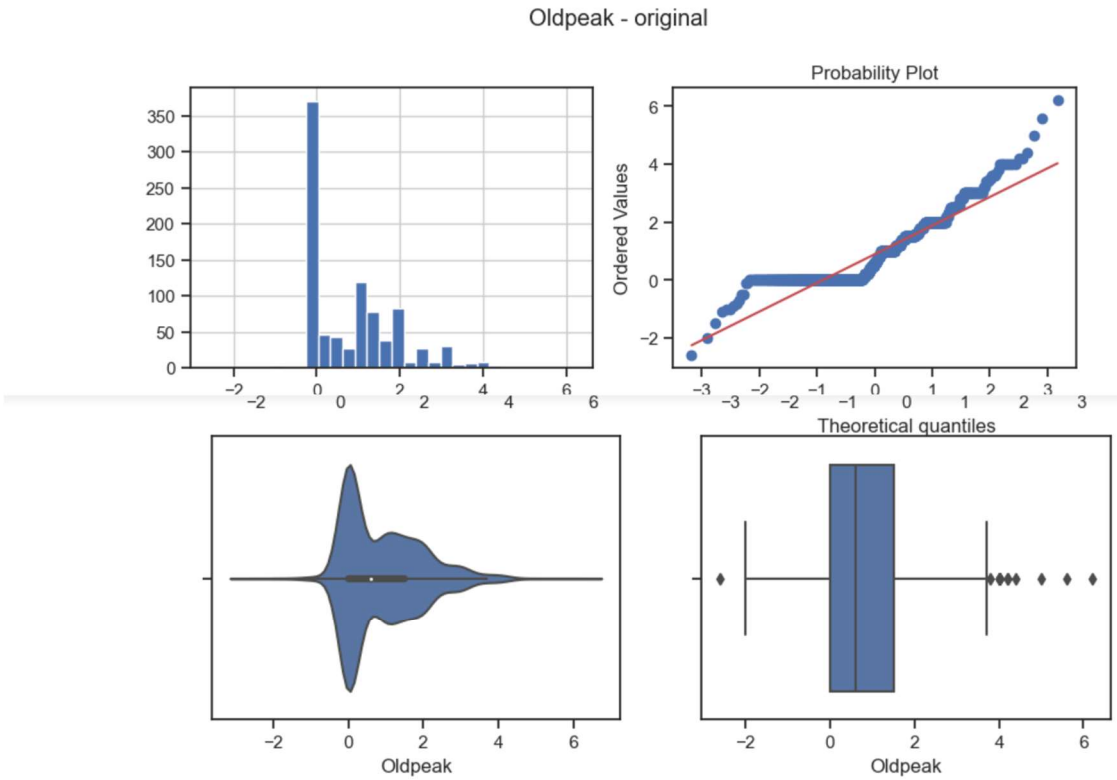
In [30]:
```
data2.describe()
```

Out[30]:

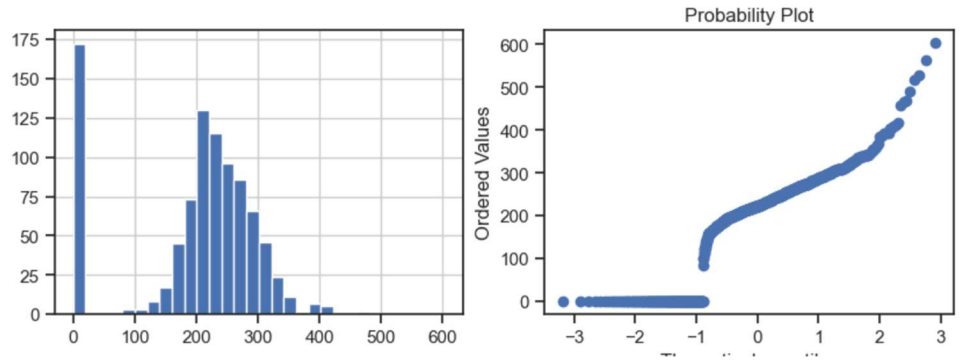
	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	917.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.803708	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.443764	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.000000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

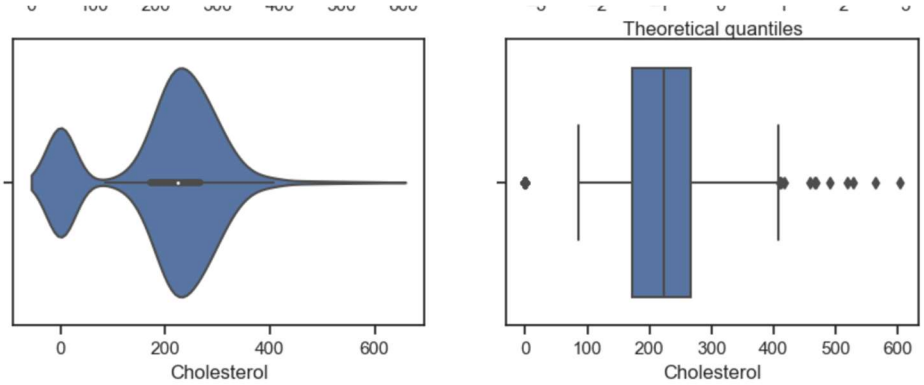
In [31]:
```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # ящик с усами
    plt.subplot(2, 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 [33]:
```
diagnostic_plots(data2, 'Oldpeak', 'Oldpeak - original')
```



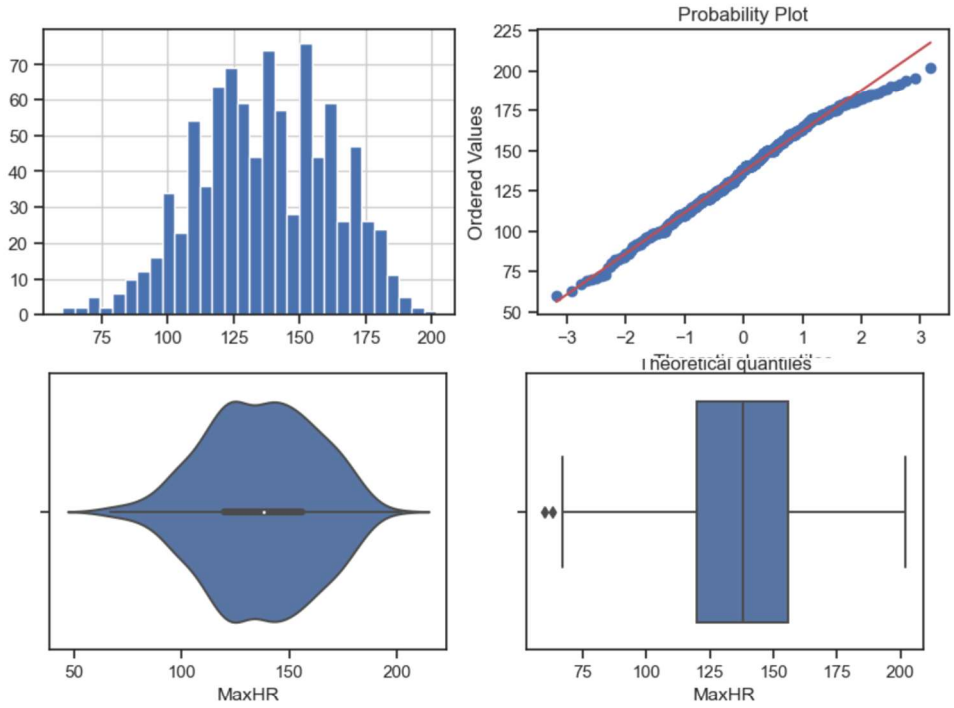
In [34]:
```
diagnostic_plots(data2, 'Cholesterol', 'Cholesterol - original')
```





In [35]:
```
diagnostic_plots(data2, 'MaxHR', 'MaxHR - original')
```

MaxHR - original



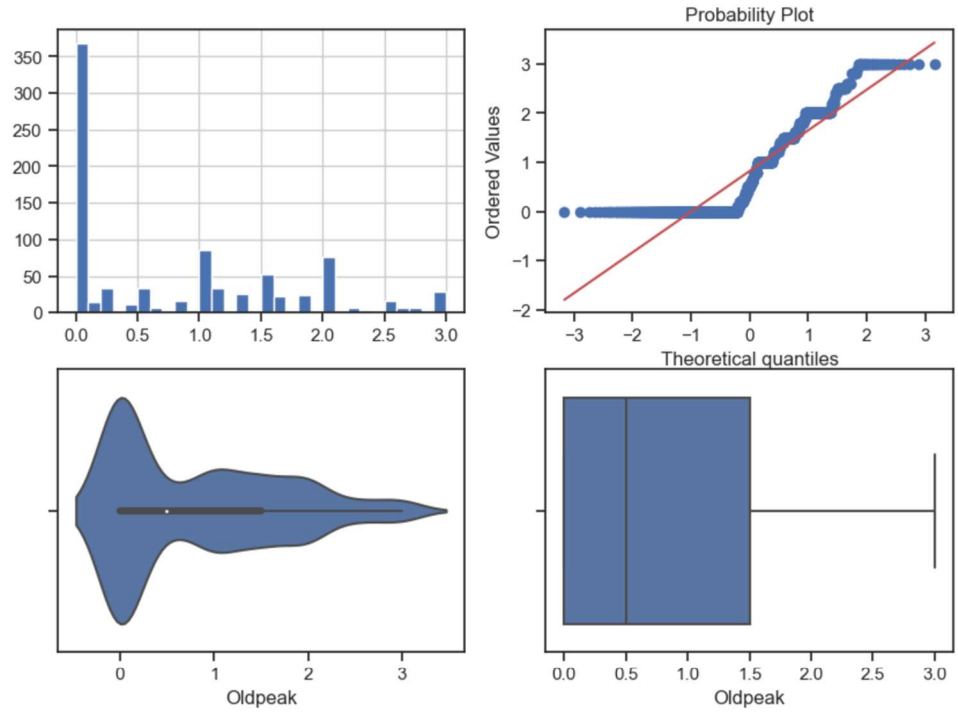
In [36]:
```
# Тип вычисления верхней и нижней границы выбросов
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
    QUANTILE = 2
    IRQ = 3
```

In [37]:
```
# 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 [39]:
```
# Вычисление верхней и нижней границы
lower_boundary, upper_boundary = get_outlier_boundaries(data2, "Oldpeak")
# Флаги для удаления выбросов
outliers_temp = np.where(data2["Oldpeak"] > upper_boundary, True,
                         np.where(data2["Oldpeak"] < lower_boundary, True, False))
# Удаление данных на основе флага
data_trimmed = data2.loc[~(outliers_temp), ]
title = 'Поле-{}, метод-{}, строк-{}'.format("Oldpeak", "QUANTILE", data_trimmed.shape[0])
diagnostic_plots(data_trimmed, "Oldpeak", title)
```

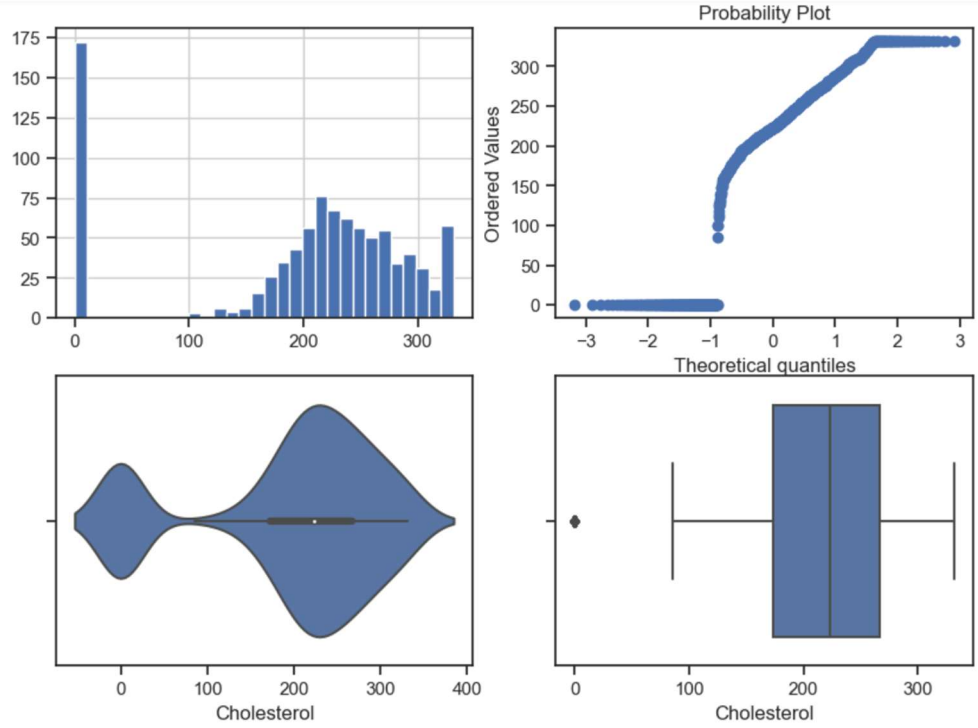
Поле-Oldpeak, метод-QUANTILE, строк-877



Замена выбросов

In [41]:
```
# Вычисление верхней и нижней границы
lower_boundary, upper_boundary = get_outlier_boundaries(data2, "Cholesterol")
# Изменение данных
data2["Cholesterol"] = np.where(data2["Cholesterol"] > upper_boundary, upper_boundary,
                       np.where(data2["Cholesterol"] < lower_boundary, lower_boundary, data2["Cholesterol"]))
title = 'Поле-{}, метод-{}'.format("Cholesterol", "QUANTILE")
diagnostic_plots(data2, "Cholesterol", title)
```

Поле-Cholesterol, метод-QUANTILE



Обработка нестандартного признака

In [42]:
```
data2.dtypes
```

Out[42]:
```
Age                int64
Sex               object
ChestPainType     object
RestingBP          int64
Cholesterol      float64
FastingBS          int64
RestingECG        object
MaxHR              int64
ExerciseAngina    object
Oldpeak          float64
ST_Slope          object
HeartDisease       int64
```

```
ExerciseAngina     object
Oldpeak           float64
ST_Slope           object
HeartDisease        int64
dtype: object
```

In [46]:
```
data2["ST_Slope_test"] = data2.apply(lambda x: pd.to_numeric(x["ST_Slope"], errors='ignore'), axis=1)
```

In [47]:
```
data2.head(5)
```

Out[47]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease	last_review_date	S
0	40	M	ATA	140	289.0	0	Normal	172	N	0.0	Up	0	Up	
1	49	F	NAP	160	180.0	0	Normal	156	N	1.0	Flat	1	Flat	
2	37	M	ATA	130	283.0	0	ST	98	N	0.0	Up	0	Up	
3	48	F	ASY	138	214.0	0	Normal	108	Y	1.5	Flat	1	Flat	
4	54	M	NAP	150	NaN	0	Normal	122	N	0.0	Up	0	Up	

In [48]:
```
data2.dtypes
```

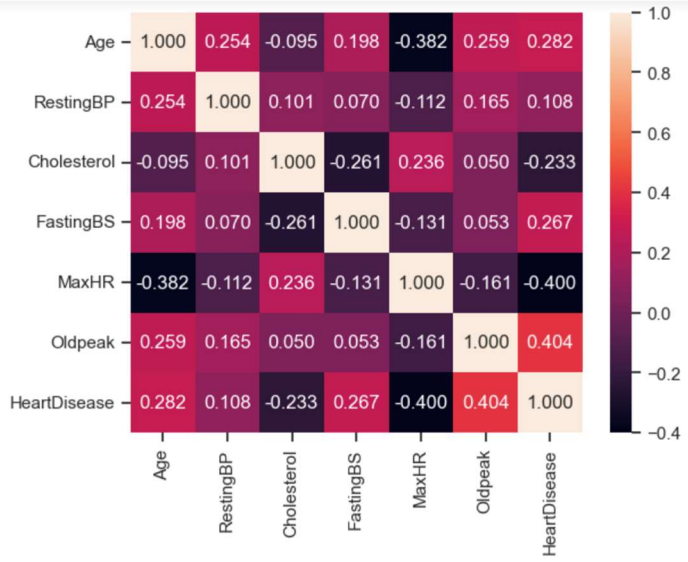
Out[48]:
```
Age                 int64
Sex                object
ChestPainType      object
RestingBP           int64
Cholesterol       float64
FastingBS           int64
RestingECG         object
MaxHR               int64
ExerciseAngina     object
Oldpeak           float64
ST_Slope           object
HeartDisease        int64
last_review_date   object
ST_Slope_test      object
dtype: object
```

Отбор признаков

Метод фильтрации (Корреляция признаков)

In [49]:
```
sns.heatmap(data.corr(), annot=True, fmt='.3f')
```

Out[49]:
```
<AxesSubplot:>
```



In [50]:
```
# 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
```

In [51]:
```
# 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
```

```
corr_groups(make_corr_df(data))
```

Out[52]:
```
[['Oldpeak', 'MaxHR', 'HeartDisease'], ['MaxHR', 'Age']]
```

Метод из группы методов вложений

In [54]:
```
!pip install scipy
```

```
Requirement already satisfied: scipy in ./opt/anaconda3/lib/python3.9/site-packages (1.9.1)
Requirement already satisfied: numpy<1.25.0,>=1.18.5 in ./opt/anaconda3/lib/python3.9/site-packages (from sci
py) (1.21.5)
```

In [56]:
```
!pip install sklearn
```

```
Collecting sklearn
  Downloading sklearn-0.0.post1.tar.gz (3.6 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: sklearn
  Building wheel for sklearn (setup.py) ... done
  Created wheel for sklearn: filename=sklearn-0.0.post1-py3-none-any.whl size=2936 sha256=a38f9208f1e16f1b54e
3c2f67f2f94e728b04e1005e1b909a737c89c1991fc50
  Stored in directory: /Users/irinaelkhimova/Library/Caches/pip/wheels/f8/e0/3d/9d0c2020c44a519b9f02ab4fa6d2a
4a996c98d79ab2f569fa1
Successfully built sklearn
Installing collected packages: sklearn
Successfully installed sklearn-0.0.post1
```

In [59]:
```
from sklearn.linear_model import LogisticRegression
# 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_
```

Out[59]:
```
array([[0.06569315, 0.43902237, 0.37652013, 0.00437418, 0.01519437]])
```

In [60]:
```
# 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()
```

Out[60]:
```
array([ True,  True,  True,  True,  True])
```