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

Лабораторная работа №3
по дисциплине «Методы машинного обучения»
«Обработка признаков (часть 2)»

ИСПОЛНИТЕЛЬ:

Цветкова Алена
Группа ИУ5-21М

____

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

____

Москва, 2021

Цель лабораторной работы: изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

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")
import scipy.stats as stats
```

In [35]:

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', -1)
```

In [36]:

```
data = pd.read_csv('archive/winemag-data-130k-v2.csv')
data.head()
```

Out[36]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle
0	0	Italy	Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity.	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe
1	1	Portugal	This is ripe and fruity, a wine that is smooth while still structured. Firm tannins are filled out with juicy red berry fruits and freshened with acidity. It's already drinkable, although it will certainly be better from 2016.	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger
2	2	US	Tart and snappy, the flavors of lime flesh and rind dominate. Some green pineapple pokes through, with crisp acidity underscoring	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle
			underscoring the flavors. The wine was all stainless-steel fermented.								
3	3	US	Pineapple rind, lemon pith and orange blossom start off the aromas. The palate is a bit more opulent, with notes of honey-drizzled guava and mango giving way to a slightly astringent, semidry finish.	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN
4	4	US	Much like the regular bottling from 2012, this comes across as rather rough and tannic, with rustic, earthy, herbal characteristics. Nonetheless, if you think of it as a pleasantly unfussy country wine, it's a good companion to a hearty winter stew.	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine

1. Масштабирование признаков

1.1 Масштабирование данных на основе Z-оценки

In [3]:

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

In [4]:

```
data.describe()
```

Out[4]:

	Unnamed: 0	points	price
count	129971.000000	129971.000000	120975.000000
mean	64985.000000	88.447138	35.363389
std	37519.540256	3.039730	41.022218
min	0.000000	80.000000	4.000000
25%	32492.500000	86.000000	17.000000
50%	64985.000000	88.000000	25.000000
75%	97477.500000	91.000000	42.000000
max	129970.000000	100.000000	3300.000000

In [5]:

```
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns = ['price'])
    return res
```

In [6]:

```
# Обучаем StandardScaler на всей выборке и масштабируем
cs11 = StandardScaler()
data_cs11_scaled_temp = cs11.fit_transform(data[['price']])
# формируем DataFrame на основе массива
data_cs11_scaled =  arr_to_df(data_cs11_scaled_temp)
data_cs11_scaled.head()
```

Out[6]:

	price
0	NaN
1	-0.496401
2	-0.520778
3	-0.545155
4	0.722456

In [7]:

```
data_cs11_scaled.describe()
```

Out[7]:

	price
count	1.209750e+05
mean	-1.442932e-15
std	1.000004e+00
min	-7.645496e-01
25%	-4.476468e-01
50%	-2.526297e-01
75%	1.617816e-01
max	7.958249e+01

In [8]:

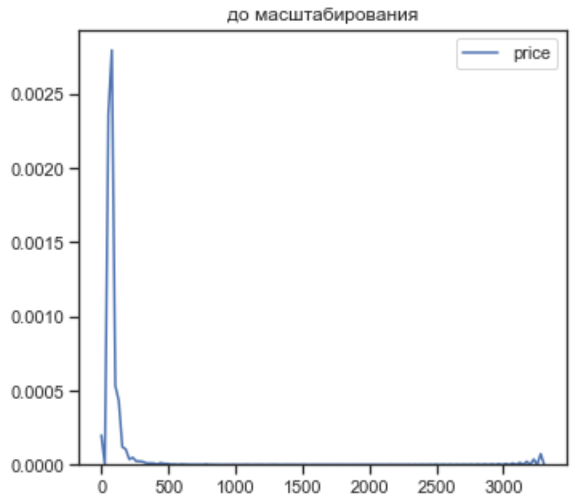
```
def draw_kde(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(
        ncols=2, figsize=(12, 5))
    # первый график
    ax1.set_title(label1)
    sns.kdeplot(data=df1[col_list], ax=ax1)
    # второй график
    ax2.set_title(label2)
    sns.kdeplot(data=df2[col_list], ax=ax2)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

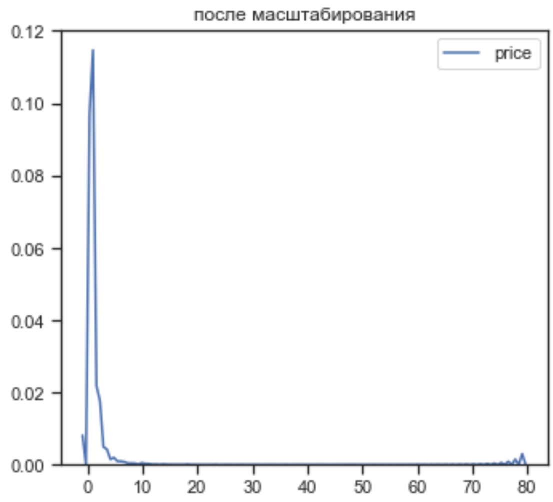
In [9]:

```
draw_kde('price', data, data_cs11_scaled, 'до масштабирования', 'после масштабирования')
```

```
//anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py:447: RuntimeWarning: invalid v
```

```
alue encountered in greater
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
//anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py:447: RuntimeWarning: invalid v
alue encountered in less
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
```





1.2 Масштабирование "Mean Normalisation"

In [10]:

```
data_price = data [['price']]
```

In [12]:

```
class MeanNormalisation:

    def fit(self, param_df):
        self.means = data_price.mean(axis=0)
        maxs = data_price.max(axis=0)
        mins = data_price.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 [13]:

```
sc21 = MeanNormalisation()
data_cs21_scaled = sc21.fit_transform(data_price)
data_cs21_scaled.describe()
```

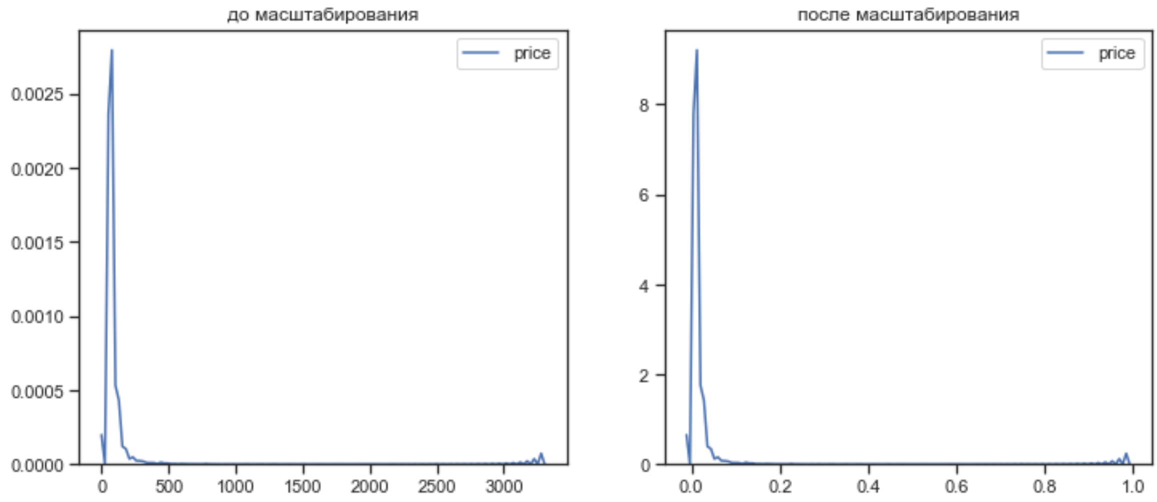
Out[13]:

	price
count	1.209750e+05
mean	-6.961787e-18
std	1.244606e-02
min	-9.515591e-03
25%	-5.571417e-03

	price
50%	-3.144232e-03
75%	2.013535e-03
max	9.904844e-01

In [14]:

```
draw_kde('price', data, data_cs21_scaled, 'до масштабирования', 'после масштабирования')
```



1.3 MinMax-масштабирование

In [16]:

```
cs31 = MinMaxScaler()
data_cs31_scaled_temp = cs31.fit_transform(data[['price']])
# формируем DataFrame на основе массива
data_cs31_scaled = arr_to_df(data_cs31_scaled_temp)
data_cs31_scaled.describe()
```

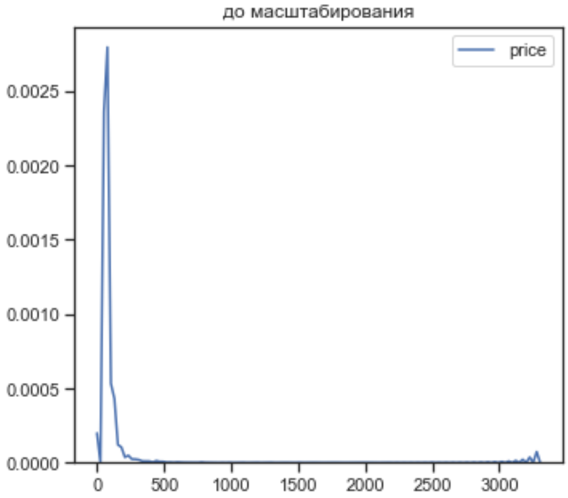
Out[16]:

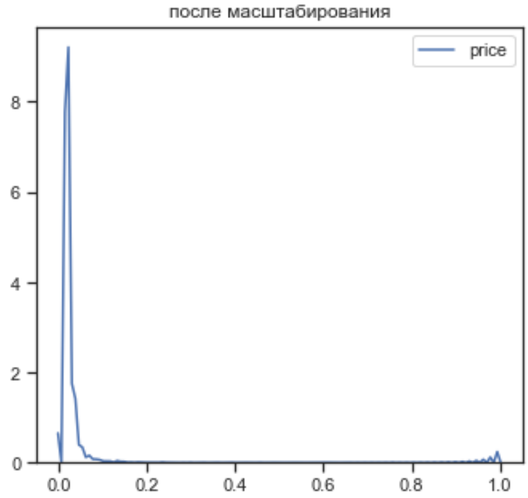
	price
count	120975.000000
mean	0.009516
std	0.012446
min	0.000000
25%	0.003944
50%	0.006371
75%	0.011529
max	1.000000

In [17]:

```
draw_kde('price', data, data_cs31_scaled, 'до масштабирования', 'после масштабирования')
```

```
//anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py:447: RuntimeWarning: invalid v
alue encountered in greater
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
//anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py:447: RuntimeWarning: invalid v
alue encountered in less
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
```





2. Обработка выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов)

2.1 Удаление выбросов

In [18]:

```
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
    QUANTILE = 2
    IRQ = 3
```

In [21]:

```
def get_outlier_boundaries(df, col, outlier_boundary_type: OutlierBoundaryType):
    if outlier_boundary_type == OutlierBoundaryType.SIGMA:
        K1 = 3
        lower_boundary = df[col].mean() - (K1 * df[col].std())
        upper_boundary = df[col].mean() + (K1 * df[col].std())

    elif outlier_boundary_type == OutlierBoundaryType.QUANTILE:
        lower_boundary = df[col].quantile(0.05)
        upper_boundary = df[col].quantile(0.95)

    elif outlier_boundary_type == OutlierBoundaryType.IRQ:
        K2 = 1.5
        IQR = df[col].quantile(0.75) - df[col].quantile(0.25)
        lower_boundary = df[col].quantile(0.25) - (K2 * IQR)
        upper_boundary = df[col].quantile(0.75) + (K2 * IQR)

    else:
        raise NameError('Unknown Outlier Boundary Type')

    return lower_boundary, upper_boundary
```

In [24]:

```
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 [28]:

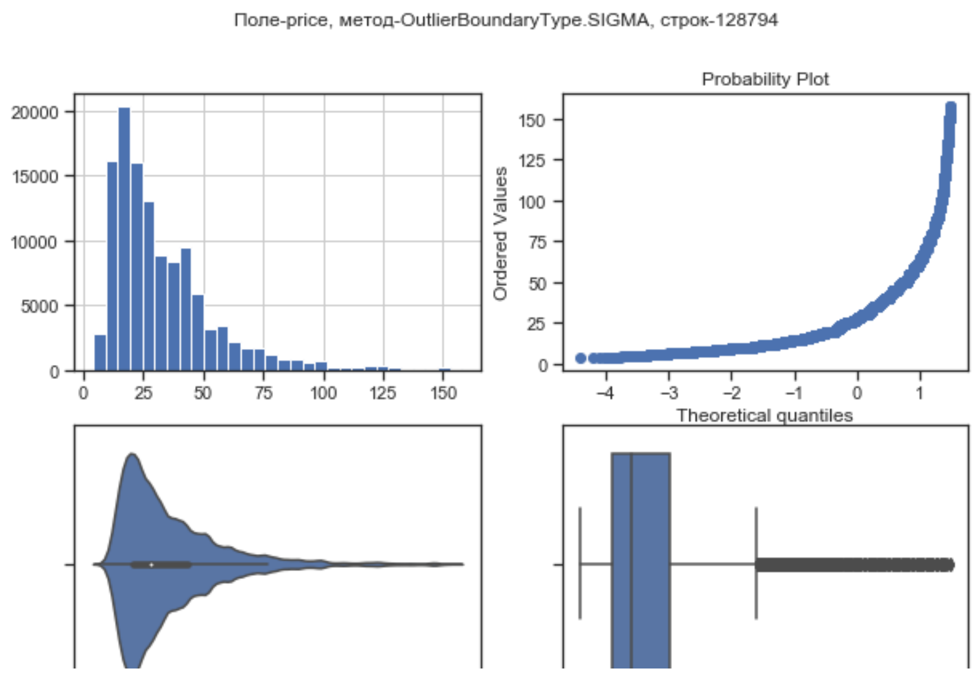
```
data[['price']].describe()
```

Out[28]:

	price
count	120975.000000
mean	35.363389
std	41.022218
min	4.000000
25%	17.000000
50%	25.000000
75%	42.000000
max	3300.000000

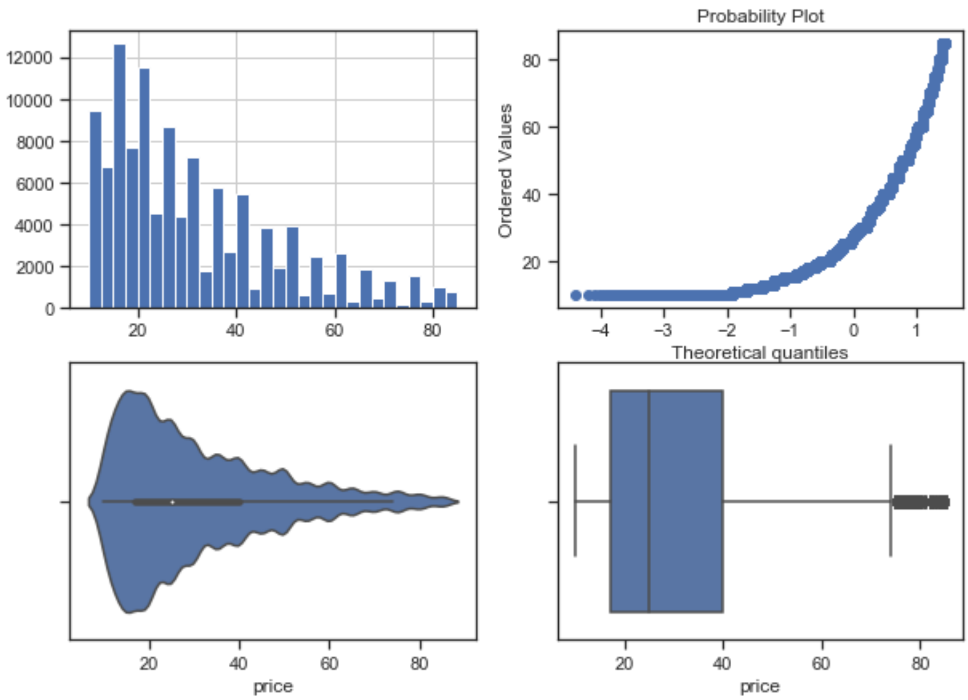
In [27]:

```
x_col_list = ['price']
for col in x_col_list:
    for obt in OutlierBoundaryType:
        # Вычисление верхней и нижней границы
        lower_boundary, upper_boundary = get_outlier_boundaries(data, col, obt)
        # Флаги для удаления выбросов
        outliers_temp = np.where(data[col] > upper_boundary, True,
                                 np.where(data[col] < lower_boundary, True, False))
        # Удаление данных на основе флага
        data_trimmed = data.loc[~(outliers_temp), ]
        title = 'Поле-{}, метод-{}, строк-{}'.format(col, obt, data_trimmed.shape[0])
        diagnostic_plots(data_trimmed, col, title)
        print(data_trimmed.describe())
```



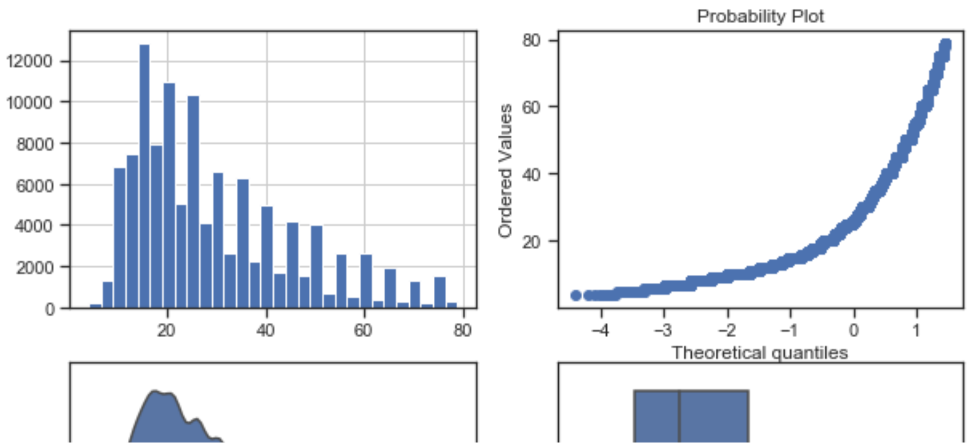
	Unnamed: 0	points	price
count	128794.000000	128794.000000	119798.000000
mean	64979.675326	88.398442	32.816174
std	37516.515114	2.999406	22.955063
min	0.000000	80.000000	4.000000
25%	32480.250000	86.000000	17.000000
50%	64975.500000	88.000000	25.000000
75%	97469.750000	91.000000	41.000000
max	129970.000000	100.000000	158.000000

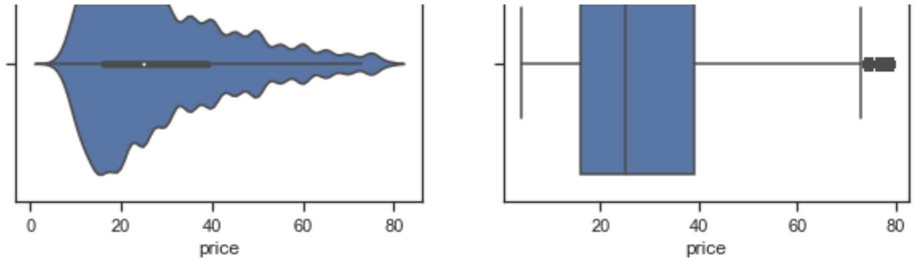
Поле-price, метод-OutlierBoundaryType.QUANTILE, строк-121588



	Unnamed: 0	points	price
count	121588.000000	121588.000000	112592.000000
mean	64914.410065	88.351581	30.442873
std	37490.776145	2.901973	17.145386
min	0.000000	80.000000	10.000000
25%	32405.750000	86.000000	17.000000
50%	64925.500000	88.000000	25.000000
75%	97379.250000	90.000000	40.000000
max	129970.000000	100.000000	85.000000

Поле-price, метод-OutlierBoundaryType.IRQ, строк-122730





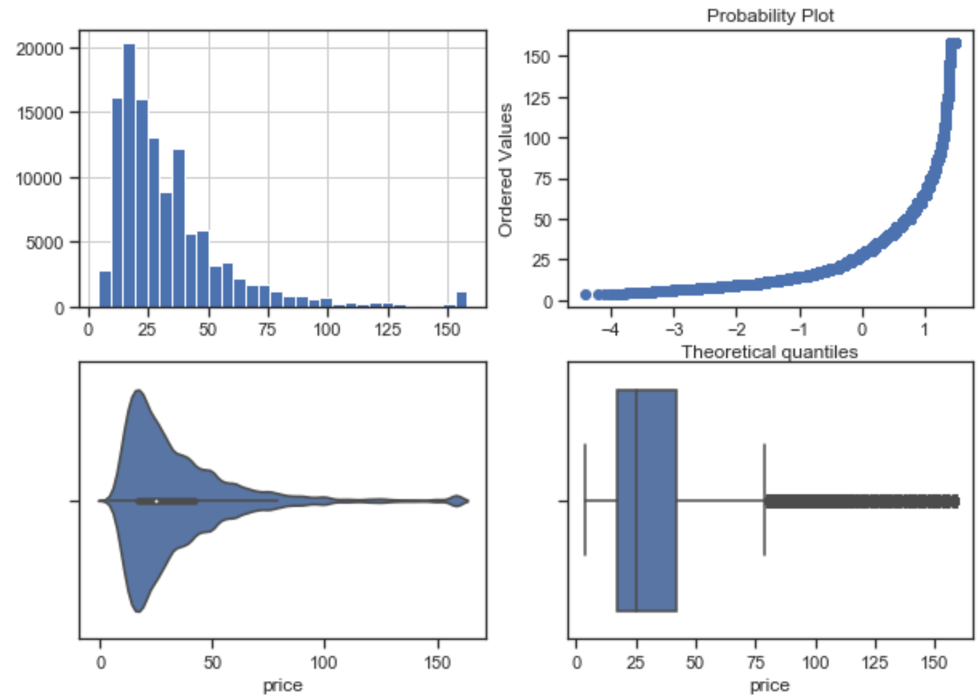

```
         Unnamed: 0         points          price
count  122730.000000  122730.000000  113734.000000
mean    64940.949295      88.225519      29.113695
std     37508.993196       2.913037      16.173909
min         0.000000      80.000000       4.000000
25%     32404.250000      86.000000      16.000000
50%     64940.500000      88.000000      25.000000
75%     97400.750000      90.000000      39.000000
max    129970.000000      99.000000      79.000000
```

2.2 Замена выбросов

In [29]:

```
for col in x_col_list:
    for obt in OutlierBoundaryType:
        # Вычисление верхней и нижней границы
        lower_boundary, upper_boundary = get_outlier_boundaries(data, col, obt)
        # Изменение данных
        data[col] = np.where(data[col] > upper_boundary, upper_boundary,
                             np.where(data[col] < lower_boundary, lower_boundary, data[col]))
        title = 'Поле-{}, метод-{}'.format(col, obt)
        diagnostic_plots(data, col, title)
        print(data_trimmed.describe())
```

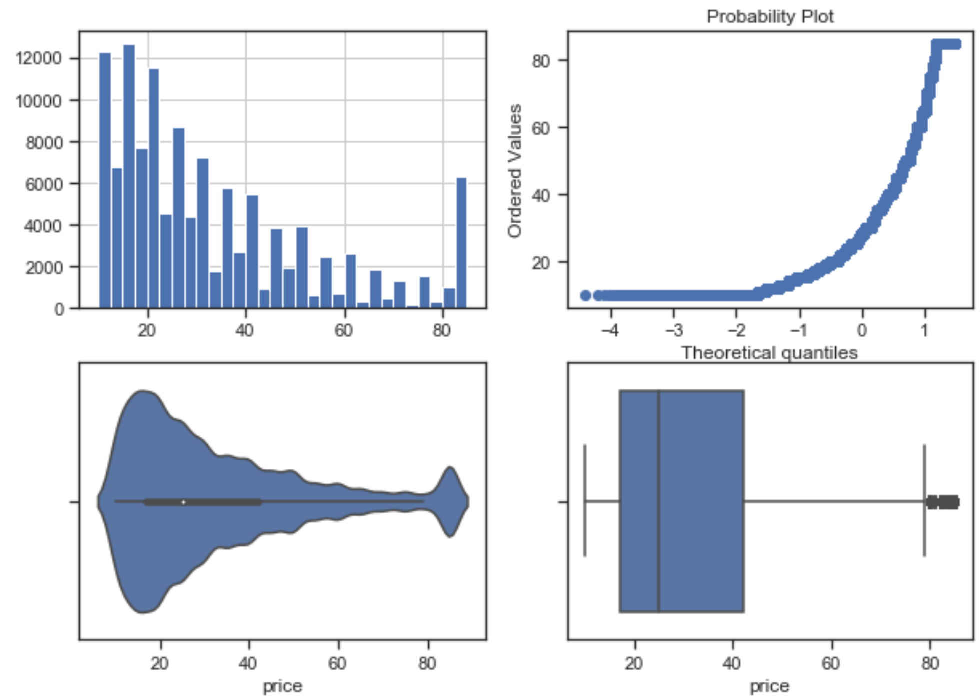
Поле-price, метод-OutlierBoundaryType.SIGMA




```
         Unnamed: 0         points          price
count  122730.000000  122730.000000  113734.000000
mean    64940.949295      88.225519      29.113695
std     37508.993196       2.913037      16.173909
min         0.000000      80.000000       4.000000
```

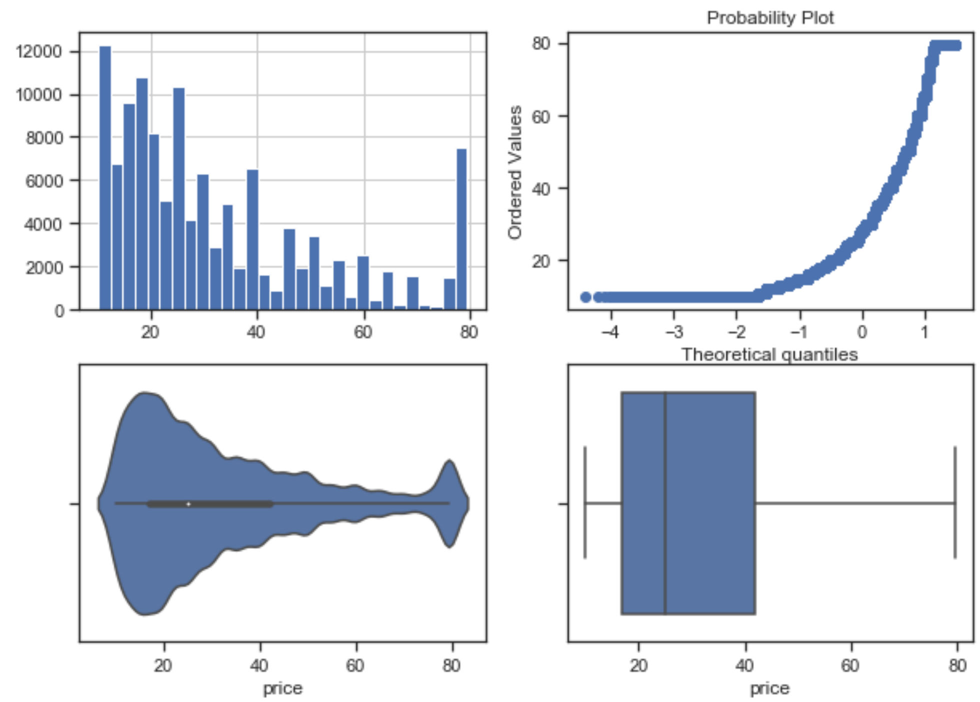
```
25%     32404.250000        86.000000      16.000000
50%     64940.500000        88.000000      25.000000
75%     97400.750000        90.000000      39.000000
max    129970.000000        99.000000      79.000000
```

Поле-price, метод-OutlierBoundaryType.QUANTILE




```
          Unnamed: 0           points          price
count  122730.000000    122730.000000  113734.000000
mean    64940.949295        88.225519      29.113695
std     37508.993196         2.913037      16.173909
min         0.000000        80.000000       4.000000
25%     32404.250000        86.000000      16.000000
50%     64940.500000        88.000000      25.000000
75%     97400.750000        90.000000      39.000000
max    129970.000000        99.000000      79.000000
```

Поле-price, метод-OutlierBoundaryType.IRQ



```
       Unnamed: 0         points          price
count  122730.000000  122730.000000  113734.000000
mean    64940.949295      88.225519      29.113695
std     37508.993196       2.913037      16.173909
min         0.000000      80.000000       4.000000
25%     32404.250000      86.000000      16.000000
50%     64940.500000      88.000000      25.000000
75%     97400.750000      90.000000      39.000000
max    129970.000000      99.000000      79.000000
```

3. Обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным)

In [42]:

```
data[['description']].head(20)
```

Out[42]:

	description
0	Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity.
1	This is ripe and fruity, a wine that is smooth while still structured. Firm tannins are filled out with juicy red berry fruits and freshened with acidity. It's already drinkable, although it will certainly be better from 2016.
2	Tart and snappy, the flavors of lime flesh and rind dominate. Some green pineapple pokes through, with crisp acidity underscoring the flavors. The wine was all stainless-steel fermented.
3	Pineapple rind, lemon pith and orange blossom start off the aromas. The palate is a bit more opulent, with notes of honey-drizzled guava and mango giving way to a slightly astringent, semidry finish.
4	Much like the regular bottling from 2012, this comes across as rather rough and tannic, with rustic, earthy, herbal characteristics. Nonetheless, if you think of it as a pleasantly unfussy country wine, it's a good companion to a hearty winter stew.
5	Blackberry and raspberry aromas show a typical Navarran whiff of green herbs and, in this case, horseradish. In the mouth, this is fairly full bodied, with tomatoey acidity. Spicy, herbal flavors complement dark plum fruit, while the finish is fresh but grabby.
6	Here's a bright, informal red that opens with aromas of candied berry, white pepper and savory herb that carry over to the palate. It's balanced with fresh acidity and soft tannins.
7	This dry and restrained wine offers spice in profusion. Balanced with acidity and a firm texture, it's very much for food.
8	Savory dried thyme notes accent sunnier flavors of preserved peach in this brisk, off-dry wine. It's fruity and fresh, with an elegant, sprightly footprint.
9	This has great depth of flavor with its fresh apple and pear fruits and touch of spice. It's off dry while balanced with acidity and a crisp texture. Drink now.
10	Soft, supple plum envelopes an oaky structure in this Cabernet, supported by 15% Merlot. Coffee and chocolate complete the picture, finishing strong at the end, resulting in a value-priced wine of attractive flavor and immediate accessibility.
11	This is a dry wine, very spicy, with a tight, taut texture and strongly mineral character layered with citrus as well as pepper. It's a food wine with its almost crisp aftertaste.
12	Slightly reduced, this wine offers a chalky, tannic backbone to an otherwise juicy explosion of rich black cherry, the whole accented throughout by firm oak and cigar box.
13	This is dominated by oak and oak-driven aromas that include roasted coffee bean, espresso, coconut and vanilla that carry over to the palate, together with plum and chocolate. Astringent, drying tannins give it a rather abrupt finish.
14	Building on 150 years and six generations of winemaking tradition, the winery trends toward a leaner style, with the classic California buttercream aroma cut by tart green apple. In this good everyday sipping wine, flavors that range from pear to barely ripe pineapple prove approachable but not distinctive.
15	Zesty orange peels and apple notes abound in this sprightly, mineral-toned Riesling. Off dry on the palate, yet racy and lean, it's a refreshing, easy quaffer with wide appeal.
16	Baked plum, molasses, balsamic vinegar and cheesy oak aromas feed into a palate that's braced by a bolt of acidity. A compact set of saucy red-berry and plum flavors features tobacco and peppery accents, while the finish is mildly green in flavor, with respectable weight and balance.
17	Raw black-cherry aromas are direct and simple but good. This has a juicy feel that thickens over time, with oak character and extract becoming more apparent. A flavor profile driven by dark-berry fruits and smoldering oak finishes meaty but hot.
18	Desiccated blackberry, leather, charred wood and mint aromas carry the nose on this full-bodied, tannic, heavily oaked Tinto Fino. Flavors of clove and woodspice sit on top of blackberry fruit, then hickory and other forceful oak-based aromas rise up and dominate the finish.
19	Red fruit aromas pervade on the nose, with cigar box and menthol notes riding in the back. The palate is slightly restrained on entry, but opens up to riper notes of cherry and plum specked with crushed pepper. This blend of Merlot, Cabernet Sauvignon and Cabernet Franc is approachable now and ready to be enjoyed.

In [38]:

```
def substr_in_desc(substr):
    lsubstr = substr.lower()
    return data.apply(lambda x: 1 if lsubstr in x['description'].lower() else 0, axis=1)
```

In [46]:

```
data['is_fruit'] = substr_in_desc('fruit')
data['is_dry'] = substr_in_desc('dry')
data['is_herb'] = substr_in_desc('herb')
```

In [48]:

```
data[['description', 'is_fruit', 'is_dry', 'is_herb']].head(20)
```

Out[48]:

	description	is_fruit	is_dry	is_herb
0	Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity.	1	0	1
1	This is ripe and fruity, a wine that is smooth while still structured. Firm tannins are filled out with juicy red berry fruits and freshened with acidity. It's already drinkable, although it will certainly be better from 2016.	1	0	0
2	Tart and snappy, the flavors of lime flesh and rind dominate. Some green pineapple pokes through, with crisp acidity underscoring the flavors. The wine was all stainless-steel fermented.	0	0	0
3	Pineapple rind, lemon pith and orange blossom start off the aromas. The palate is a bit more opulent, with notes of honey-drizzled guava and mango giving way to a slightly astringent, semidry finish.	0	1	0
4	Much like the regular bottling from 2012, this comes across as rather rough and tannic, with rustic, earthy, herbal characteristics. Nonetheless, if you think of it as a pleasantly unfussy country wine, it's a good companion to a hearty winter stew.	0	0	1
5	Blackberry and raspberry aromas show a typical Navarran whiff of green herbs and, in this case, horseradish. In the mouth, this is fairly full bodied, with tomatoey acidity. Spicy, herbal flavors complement dark plum fruit, while the finish is fresh but grabby.	1	0	1
6	Here's a bright, informal red that opens with aromas of candied berry, white pepper and savory herb that carry over to the palate. It's balanced with fresh acidity and soft tannins.	0	0	1
7	This dry and restrained wine offers spice in profusion. Balanced with acidity and a firm texture, it's very much for food.	0	1	0
8	Savory dried thyme notes accent sunnier flavors of preserved peach in this brisk, off-dry wine. It's fruity and fresh, with an elegant, sprightly footprint.	1	1	0
9	This has great depth of flavor with its fresh apple and pear fruits and touch of spice. It's off dry while balanced with acidity and a crisp texture. Drink now.	1	1	0
10	Soft, supple plum envelopes an oaky structure in this Cabernet, supported by 15% Merlot. Coffee and chocolate complete the picture, finishing strong at the end, resulting in a value-priced wine of attractive flavor and immediate accessibility.	0	0	0
11	This is a dry wine, very spicy, with a tight, taut texture and strongly mineral character layered with citrus as well as pepper. It's a food wine with its almost crisp aftertaste.	0	1	0
12	Slightly reduced, this wine offers a chalky, tannic backbone to an otherwise juicy explosion of rich black cherry, the whole accented throughout by firm oak and cigar box.	0	0	0
13	This is dominated by oak and oak-driven aromas that include roasted coffee bean, espresso, coconut and vanilla that carry over to the palate, together with plum and chocolate. Astringent, drying tannins give it a rather abrupt finish.	0	1	0
14	Building on 150 years and six generations of winemaking tradition, the winery trends toward a leaner style, with the classic California buttercream aroma cut by tart green apple. In this good everyday sipping wine, flavors that range from pear to barely ripe pineapple prove approachable but not distinctive.	0	0	0
15	Zesty orange peels and apple notes abound in this sprightly, mineral-toned Riesling. Off dry on the palate, yet racy and lean, it's a refreshing, easy quaffer with wide appeal.	0	1	0
16	Baked plum, molasses, balsamic vinegar and cheesy oak aromas feed into a palate that's braced by a bolt of acidity. A compact set of saucy red-berry and plum flavors features tobacco and peppery accents, while the finish is mildly green in flavor, with respectable weight and balance.	0	0	0
17	Raw black-cherry aromas are direct and simple but good. This has a juicy feel that thickens over time, with oak character and extract becoming more apparent. A flavor profile driven by dark-berry fruits and smoldering oak finishes meaty but hot.	1	0	0
	Desiccated blackberry, leather, charred wood and mint aromas carry the nose on this full-bodied, tannic, heavily			

	description	is_fruit	is_dry	is_herb
18	oaked Tinto Fino. Flavors of clove and woodspice sit on top of blackberry fruit, then hickory and other forceful oak-based aromas rise up and dominate the finish.	1	0	0
19	Red fruit aromas pervade on the nose, with cigar box and menthol notes riding in the back. The palate is slightly restrained on entry, but opens up to riper notes of cherry and plum specked with crushed pepper. This blend of Merlot, Cabernet Sauvignon and Cabernet Franc is approachable now and ready to be enjoyed.	1	0	0

4. Отбор признаков

4.1 Один метод из группы методов фильтрации (filter methods)

In [49]:

```
data = pd.read_csv("archive/train.csv")
```

In [71]:

```
data.head()
```

Out[71]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411

In [74]:

```
data['price_range'].unique()
```

Out[74]:

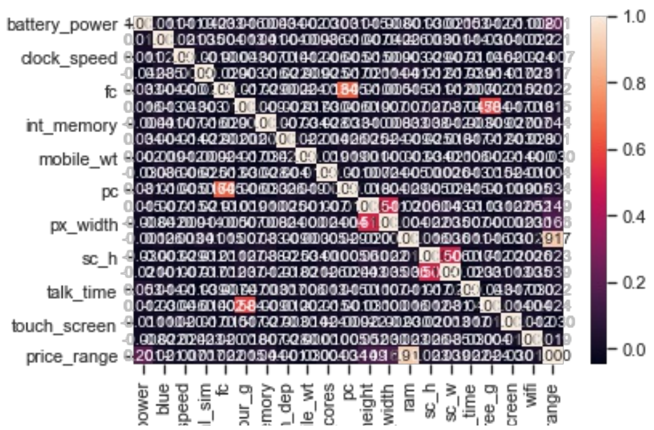
```
array([1, 2, 3, 0])
```

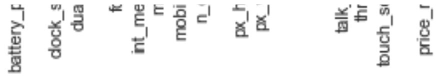
In [50]:

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

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x122587e10>
```





In [69]:

```
def make_corr_df(df):
    cr = data.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= 0.6]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
    cr.columns = ['f1', 'f2', 'corr']
    return cr
```

In [70]:

```
make_corr_df(data)
```

Out[70]:

	f1	f2	corr
0	price_range	ram	0.917046
1	ram	price_range	0.917046
2	fc	pc	0.644595
3	pc	fc	0.644595

In [80]:

```
from sklearn.feature_selection import SelectKBest, SelectPercentile
from sklearn.feature_selection import mutual_info_classif, mutual_info_regression
```

In [77]:

```
data_y = data['price_range']
data_x = data.drop(columns=['price_range'])
```

In [83]:

```
sel_mi = SelectKBest(mutual_info_classif, k=5).fit(data_x, data_y)
list(zip(data.columns, sel_mi.get_support()))
```

Out[83]:

```
[('battery_power', True),
 ('blue', False),
 ('clock_speed', False),
 ('dual_sim', False),
 ('fc', False),
 ('four_g', False),
 ('int_memory', False),
 ('m_dep', False),
 ('mobile_wt', True),
 ('n_cores', False),
 ('pc', False),
 ('px_height', True),
 ('px_width', True),
 ('ram', True),
 ('sc_h', False),
 ('sc_w', False),
 ('talk_time', False),
 ('three_g', False),
 ('touch_screen', False),
```

('wifi', False)]

4.2 Один метод из группы методов обертывания (wrapper methods)

In [97]:

```
from sklearn.neighbors import KNeighborsClassifier
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS
```

In [87]:

```
knn = KNeighborsClassifier(n_neighbors=3)
```

In [100]:

```
%%time
efs1 = EFS(knn,
           min_features=2,
           max_features=4,
           scoring='accuracy',
           print_progress=True,
           cv=5)

efs1 = efs1.fit(data_x, data_y, custom_feature_names=data_x.columns)

print('Best accuracy score: %.2f' % efs1.best_score_)
print('Best subset (indices):', efs1.best_idx_)
print('Best subset (corresponding names):', efs1.best_feature_names_)
```

```
Features: 6175/6175
```

```
Best accuracy score: 0.92
Best subset (indices): (0, 11, 12, 13)
Best subset (corresponding names): ('battery_power', 'px_height', 'px_width', 'ram')
CPU times: user 10min 47s, sys: 6.88 s, total: 10min 53s
Wall time: 11min 9s
```

4.3 Один метод из группы методов вложений (embedded methods)

In [93]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectFromModel
```

In [94]:

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

```
//anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default mu
lti_class will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[94]:

```
array([[-2.47788621e-02, -3.13672752e-02,  3.50019720e-01,
         3.85881786e-01,  6.59697944e-04,  4.99739083e-01,
        -1.93132480e-02, -2.04670182e-01,  4.62426601e-02,
```

```
       -2.02571984e-01,  6.28383749e-03, -1.52393954e-02,
       -1.39793167e-02, -4.05001109e-02,  6.07620830e-02,
       -1.48659851e-03,  6.47557627e-02,  4.38600884e-01,
        3.03652667e-01,  1.06572135e+00],
      [-1.12587937e-04,  4.78419420e-03, -6.74048339e-02,
        5.50224620e-02,  4.11502476e-03,  4.34393024e-02,
        1.03298043e-03,  3.54089093e-01,  8.22881840e-05,
       -5.79593087e-02,  1.06101552e-03,  1.58811178e-04,
       -7.99114510e-05, -5.35183081e-04,  5.09687208e-05,
       -1.18114103e-02,  1.88852967e-02, -3.91556369e-02,
        7.51943863e-02,  7.45263081e-03],
      [-7.92991568e-05, -6.82940987e-02,  6.92157866e-03,
       -1.14854089e-01,  1.68529525e-02, -2.71810945e-01,
       -5.39942783e-03, -1.83722744e-01,  3.64448401e-03,
        4.34411763e-02, -7.61445896e-03,  3.48645396e-05,
       -1.54482788e-04,  5.63417607e-04, -2.76141291e-02,
        5.70079912e-03, -2.21495110e-03,  2.53587722e-01,
       -1.61112354e-01, -4.58258568e-02],
      [ 3.03645148e-02, -1.16761749e+00,  9.25860013e-02,
        6.46306246e-01, -2.55035610e-02,  5.24634672e-01,
        7.95408196e-02,  4.69415624e-01, -9.25701520e-02,
        2.58277320e-01,  4.83048129e-02,  1.99023077e-02,
        1.81187171e-02,  5.02092555e-02,  2.12197967e-01,
        3.34951524e-02,  2.86646993e-02, -4.48650954e-01,
       -1.36171590e-01, -1.51717306e+00]])
```

In [95]:

```
sel_e_lr1 = SelectFromModel(e_lr1)
sel_e_lr1.fit(data_x, data_y)
sel_e_lr1.get_support()
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default mu
lti_class will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[95]:

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

In [ ]: