осковский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №3

по дисциплине «Методы машинного обучения»

на тему«Обработка признаков (часть 2)»

Выполнил:

студент группы： ИУ5-23М

Ся Тунтун

Москва — 2022 г.

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

Задание: Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных.

Для выполнения следующих пунктов можно использовать несколько различных наборов данных

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 pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import datetime
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MaxAbsScaler
import warnings
warnings.simplefilter("ignore", UserWarning)
```

In [2]:
```
dataset = pd.read_csv('phonetrain.csv')
```

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

Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cor
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

5 rows × 21 columns

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

Out[4]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000

8 rows × 21 columns

Разделим выборку на обучающую и тестовую

In [5]:
```
X=dataset.drop('price_range',axis=1)
```

In [6]:
```
y=dataset['price_range']
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state
```

Масштабирование

Функция для восстановления датафрейма

на основе масштабированных данных

In [8]:
```
# Function for restoring the dataframe
# based on scaled data

def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=X.columns)
    return res
```

Преобразуем массивы в DataFrame

In [9]:
```
# Converting arrays to 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[9]: ((1340, 20), (660, 20))

Масштабирование данных на основе Z-оценки

Обучаем StandardScaler на всей выборке и масштабируем

In [10]:
```
# Train the StandardScaler on the entire sample and scale it
cs11 = StandardScaler()
data_cs11_scaled_temp = cs11.fit_transform(X)
# forming a DataFrame based on an array
data_cs11_scaled =  arr_to_df(data_cs11_scaled_temp)
data_cs11_scaled
```

Out[10]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_de
0	-0.902597	-0.990050	0.830779	-1.019184	-0.762495	-1.043966	-1.380644	0.34074
1	-0.495139	1.010051	-1.253064	0.981177	-0.992890	0.957886	1.155024	0.68754
2	-1.537686	1.010051	-1.253064	0.981177	-0.532099	0.957886	0.493546	1.38116
3	-1.419319	1.010051	1.198517	-1.019184	-0.992890	-1.043966	-1.215274	1.03435
4	1.325906	1.010051	-0.395011	-1.019184	2.002254	0.957886	0.658915	0.34074
...	...	...	...	...	...	...	...	.
1995	-1.011860	1.010051	-1.253064	0.981177	-0.992890	0.957886	-1.656260	1.03435
1996	1.653694	1.010051	1.321096	0.981177	-0.992890	-1.043966	0.383299	-1.04649
1997	1.530773	-0.990050	-0.762748	0.981177	-0.762495	0.957886	0.217930	0.68754
1998	0.622527	-0.990050	-0.762748	-1.019184	-0.071307	0.957886	0.769162	-1.39330
1999	-1.658331	1.010051	0.585621	0.981177	0.159088	0.957886	0.714039	1.38116

2000 rows × 20 columns

In [11]:
```
data_cs11_scaled.describe()
```

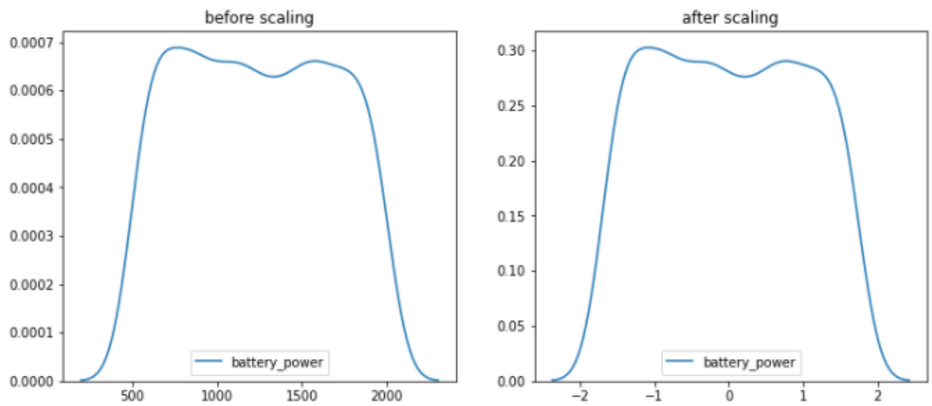
Out[11]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	
count	2.000000e+03	2.000000e+03	2.000000e+03	2.000000e+03	2.000000e+03	2.000000e+03	2.
mean	2.128298e-16	-1.927347e-16	-2.172151e-16	3.990142e-16	9.230117e-17	-2.048361e-16	-8
std	1.000250e+00	1.000250e+00	1.000250e+00	1.000250e+00	1.000250e+00	1.000250e+00	1.
min	-1.678817e+00	-9.900495e-01	-1.253064e+00	-1.019184e+00	-9.928904e-01	-1.043966e+00	-1.
25%	-8.804033e-01	-9.900495e-01	-1.007906e+00	-1.019184e+00	-7.624947e-01	-1.043966e+00	-8
50%	-2.849593e-02	-9.900495e-01	-2.727384e-02	9.811771e-01	-3.017032e-01	9.578860e-01	-2
75%	8.575560e-01	1.010051e+00	8.307794e-01	9.811771e-01	6.198797e-01	9.578860e-01	8
max	1.728812e+00	1.010051e+00	1.811412e+00	9.811771e-01	3.384628e+00	9.578860e-01	1.

Построение плотности распределения

In [12]:
```
# Plotting the distribution density
def draw_ba(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(
        ncols=2, figsize=(12, 5))
# first graph
    ax1.set_title(label1)
    sns.kdeplot(data=df1[col_list], ax=ax1)
# second graph
    ax2.set_title(label2)
    sns.kdeplot(data=df2[col_list], ax=ax2)
    plt.show()
```

In [13]:
```
draw_ba('battery_power', dataset, data_cs11_scaled, 'before scaling', 'after scaling')
```



Обучаем StandardScaler на обучающей выборке и масштабируем обучающую и тестовую выборки

```
In [14]: # Training the StandardScaler on the training sample
         # and scale the training and test samples
         cs12 = StandardScaler()
         cs12.fit(X_train)
         data_cs12_scaled_train_temp = cs12.transform(X_train)
         data_cs12_scaled_test_temp = cs12.transform(X_test)
         # forming a DataFrame based on an array
         data_cs12_scaled_train = arr_to_df(data_cs12_scaled_train_temp)
         data_cs12_scaled_test = arr_to_df(data_cs12_scaled_test_temp)
```

```
In [15]: data_cs12_scaled_train.describe()
```

Out[15]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	
count	1.340000e+03	1.340000e+03	1.340000e+03	1.340000e+03	1.340000e+03	1.340000e+03	1
mean	2.398579e-17	-2.656250e-16	-7.854414e-17	-1.347181e-16	-2.485574e-18	-1.413463e-16	6
std	1.000373e+00	1.000373e+00	1.000373e+00	1.000373e+00	1.000373e+00	1.000373e+00	1
min	-1.687724e+00	-9.822471e-01	-1.250218e+00	-1.005988e+00	-1.001647e+00	-1.058420e+00	-1
25%	-8.904450e-01	-9.822471e-01	-1.002540e+00	-1.005988e+00	-7.713831e-01	-1.058420e+00	-9
50%	1.072825e-02	-9.822471e-01	-1.182938e-02	9.940476e-01	-3.108559e-01	9.448044e-01	-1
75%	8.560016e-01	1.018074e+00	8.550425e-01	9.940476e-01	6.101985e-01	9.448044e-01	8
max	1.693370e+00	1.018074e+00	1.845753e+00	9.940476e-01	3.373362e+00	9.448044e-01	1

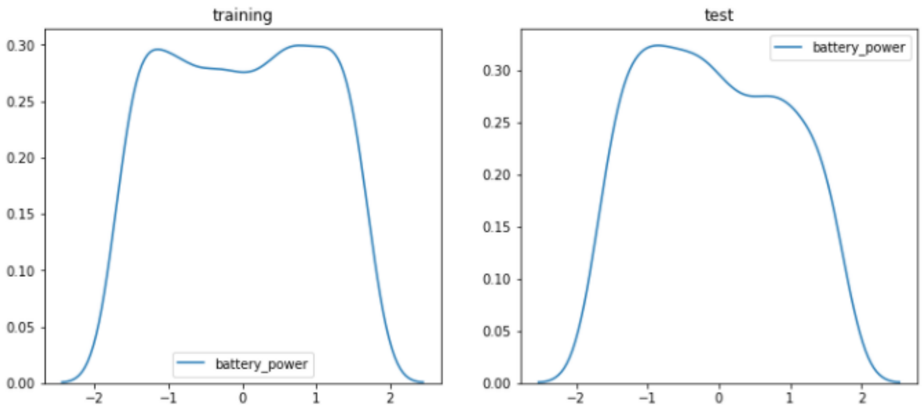
```
In [16]: data_cs12_scaled_test.describe()
```

Out[16]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	
count	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000	6
mean	-0.066604	0.023975	0.047651	0.039485	-0.028260	-0.041632	-0.045578	
std	0.975427	1.000901	1.030872	0.999742	0.998750	1.002257	1.016119	
min	-1.685465	-0.982247	-1.250218	-1.005988	-1.001647	-1.058420	-1.680111	
25%	-0.918677	-0.982247	-1.002540	-1.005988	-0.771383	-1.058420	-0.959697	
50%	-0.142855	1.018074	0.112009	0.994048	-0.310856	0.944804	-0.100742	
75%	0.749284	1.018074	0.978881	0.994048	0.610199	0.944804	0.827483	
max	1.688853	1.018074	1.845753	0.994048	3.143098	0.944804	1.755709	

распределения для обучающей и тестовой выборки немного отличаются

In [17]:
```
draw_ba('battery_power', data_cs12_scaled_train,data_cs12_scaled_test, 'training', 'te
```



Масштабирование "Mean Normalisation"

In [18]:
```
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 [19]:
```
sc21 = MeanNormalisation()
data_cs21_scaled = sc21.fit_transform(X)
data_cs21_scaled.describe()
```

Out[19]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memc
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.0000
mean	-0.006501	0.003955	0.005079	0.006515	-0.002132	-0.006858	-0.0043
std	0.293533	0.500100	0.326402	0.500035	0.228497	0.499662	0.2926
min	-0.499165	-0.491045	-0.403821	-0.502985	-0.228947	-0.528358	-0.4889
25%	-0.264863	-0.491045	-0.323821	-0.502985	-0.176316	-0.528358	-0.2631
50%	-0.014863	-0.491045	-0.003821	0.497015	-0.071053	0.471642	-0.0051
75%	0.245157	0.508955	0.276179	0.497015	0.139474	0.471642	0.2529
max	0.500835	0.508955	0.596179	0.497015	0.771053	0.471642	0.5110

In [20]:
```
cs22 = MeanNormalisation()
cs22.fit(X_train)
data_cs22_scaled_train = cs22.transform(X_train)
data_cs22_scaled_test = cs22.transform(X_test)
```

In [21]:
```
data_cs22_scaled_train.describe()
```

Out[21]:

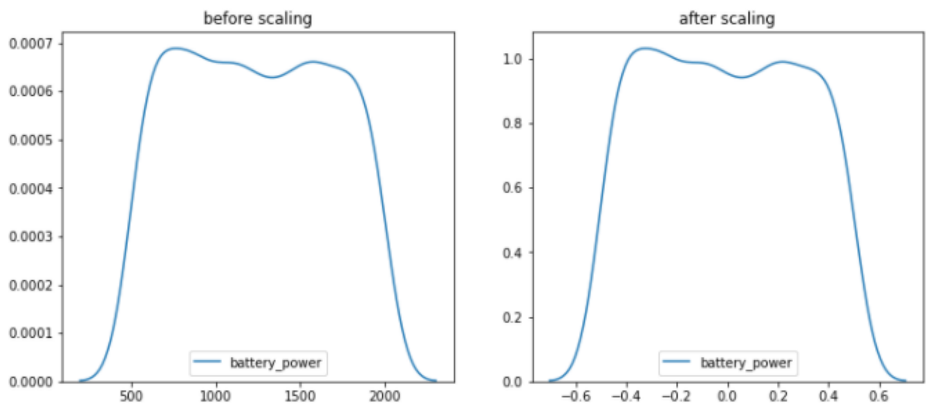
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_
count	1.340000e+03	1.340000e+03	1.340000e+03	1.340000e+03	1.340000e+03	1.340000e+03	1.340
mean	-3.894066e-18	1.617280e-16	7.224217e-16	1.062169e-16	2.961976e-17	1.149992e-16	1.93
std	2.958727e-01	5.001064e-01	3.231210e-01	5.001778e-01	2.286563e-01	4.993815e-01	2.91
min	-4.991650e-01	-4.910448e-01	-4.038209e-01	-5.029851e-01	-2.289474e-01	-5.283582e-01	-4.
25%	-2.633601e-01	-4.910448e-01	-3.238209e-01	-5.029851e-01	-1.763158e-01	-5.283582e-01	-2.
50%	3.173013e-03	-4.910448e-01	-3.820896e-03	4.970149e-01	-7.105263e-02	4.716418e-01	-5.
75%	2.531730e-01	5.089552e-01	2.761791e-01	4.970149e-01	1.394737e-01	4.716418e-01	2.52
max	5.008350e-01	5.089552e-01	5.961791e-01	4.970149e-01	7.710526e-01	4.716418e-01	5.11

In [22]:
```
data_cs22_scaled_test.describe()
```

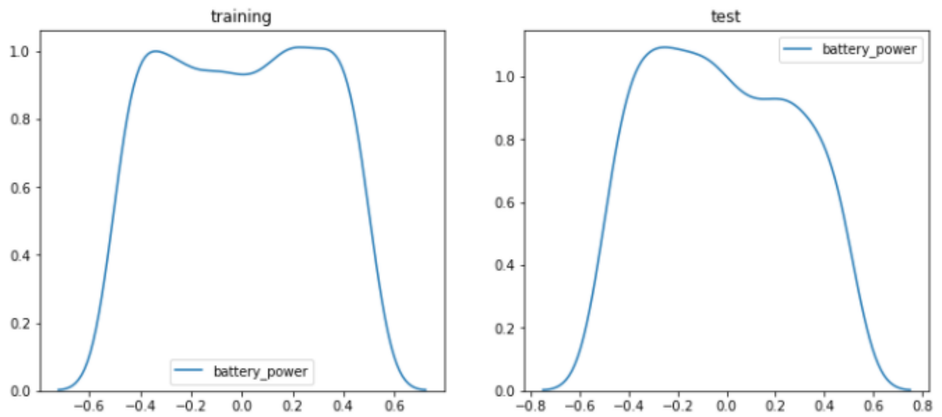
Out[22]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	
count	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000	6
mean	-0.019699	0.011986	0.015391	0.019742	-0.006459	-0.020782	-0.013265	
std	0.288495	0.500370	0.332972	0.499862	0.228285	0.500322	0.295743	
min	-0.498497	-0.491045	-0.403821	-0.502985	-0.228947	-0.528358	-0.488999	
25%	-0.271710	-0.491045	-0.323821	-0.502985	-0.176316	-0.528358	-0.279321	
50%	-0.042251	0.508955	0.036179	0.497015	-0.071053	0.471642	-0.029321	
75%	0.221610	0.508955	0.316179	0.497015	0.139474	0.471642	0.240840	
max	0.499499	0.508955	0.596179	0.497015	0.718421	0.471642	0.511001	

In [23]:
```
draw_ba('battery_power', dataset, data_cs21_scaled, 'before scaling', 'after scaling')
```



In [24]:
```
draw_ba('battery_power', data_cs22_scaled_train, data_cs22_scaled_test, 'training', 'tes
```



MinMax-масштабирование

Обучаем StandardScaler на всей выборке и масштабируем

In [25]:
```
# Train the StandardScaler on the entire sample and scale it
cs31 = MinMaxScaler()
data_cs31_scaled_temp = cs31.fit_transform(X)
# forming a DataFrame based on an array
data_cs31_scaled = arr_to_df(data_cs31_scaled_temp)
data_cs31_scaled.describe()
```

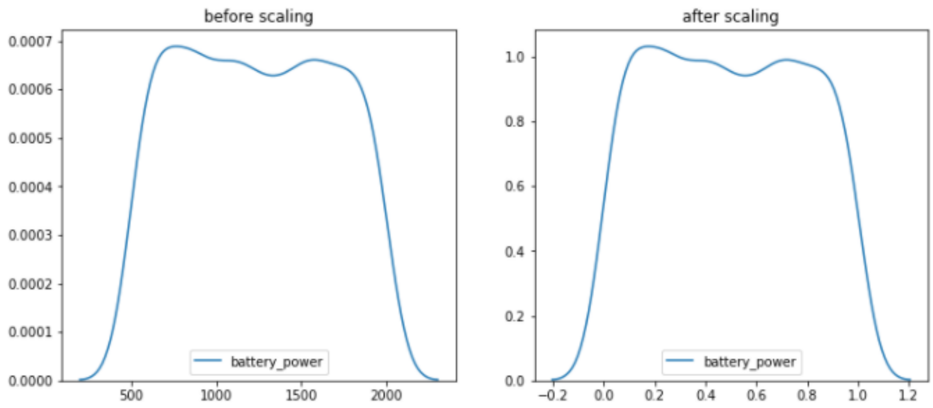
Out[25]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	0.492664	0.4950	0.408900	0.509500	0.226816	0.521500	0.484621
std	0.293533	0.5001	0.326402	0.500035	0.228497	0.499662	0.292673
min	0.000000	0.0000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.234302	0.0000	0.080000	0.000000	0.052632	0.000000	0.225806
50%	0.484302	0.0000	0.400000	1.000000	0.157895	1.000000	0.483871
75%	0.744322	1.0000	0.680000	1.000000	0.368421	1.000000	0.741935
max	1.000000	1.0000	1.000000	1.000000	1.000000	1.000000	1.000000

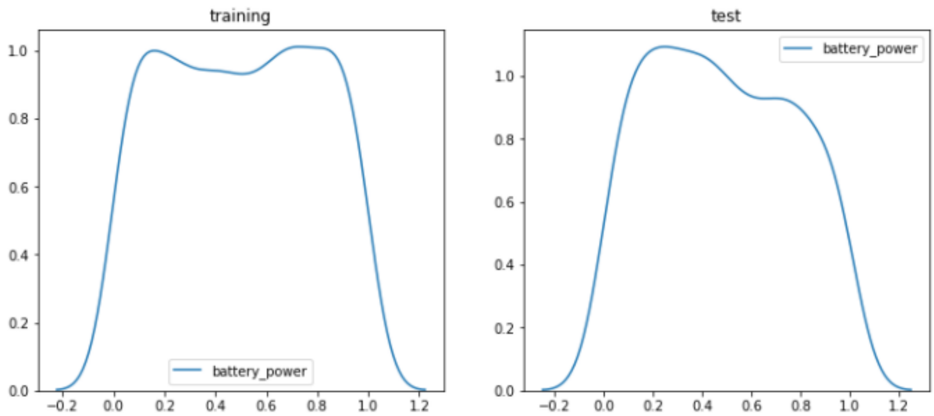
формируем DataFrame на основе массива

In [26]:
```
cs32 =  MinMaxScaler()
cs32.fit(X_train)
data_cs32_scaled_train_temp = cs32.transform(X_train)
data_cs32_scaled_test_temp = cs32.transform(X_test)
# forming a DataFrame based on an array
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 [27]: draw_ba('battery_power', dataset, data_cs31_scaled, 'before scaling', 'after scaling')
```




```
In [28]: draw_ba('battery_power',data_cs32_scaled_train, data_cs32_scaled_test,'training','test
```



Обработка выбросов

Удаление выбросов

```
In [29]: dataset2=pd.read_csv('deputies_dataset.csv')
```

```
In [30]: dataset2.head()
```

Out[30]:

	bugged_date	receipt_date	deputy_id	political_party	state_code	deputy_name	receipt_social_s
0	0	2013-03-27 00:00:00	1772	PSB	SP	Abelardo Camarinha	
1	0	2013-07-24 00:00:00	1772	PSB	SP	Abelardo Camarinha	
2	0	2013-02-17 00:00:00	1772	PSB	SP	Abelardo Camarinha	
3	0	2013-03-15 00:00:00	1772	PSB	SP	Abelardo Camarinha	
4	0	2013-01-27 00:00:00	1772	PSB	SP	Abelardo Camarinha	

In [31]:
```
dataset2.describe()
```

Out[31]:

	bugged_date	deputy_id	receipt_social_security_number	receipt_value
count	3.014902e+06	3.014902e+06	2.493950e+06	3.014902e+06
mean	1.642873e-02	1.869101e+03	1.372664e+13	5.791575e+02
std	1.271174e-01	7.014751e+02	2.057245e+13	1.925418e+03
min	0.000000e+00	1.200000e+01	0.000000e+00	0.000000e+00
25%	0.000000e+00	1.467000e+03	2.087236e+12	5.000000e+01
50%	0.000000e+00	1.882000e+03	7.423935e+12	1.420000e+02
75%	0.000000e+00	2.340000e+03	1.123836e+13	4.720000e+02
max	1.000000e+00	3.173000e+03	9.874986e+13	2.150000e+05

In [32]:
```
dataset2.shape
```

Out[32]: (3014902, 10)

Функция построения графиков - ящики с усами

In [34]:
```
# Drawing function-Box diagram
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(15,7))
    # Box diagram
    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])
    # Box diagram
    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

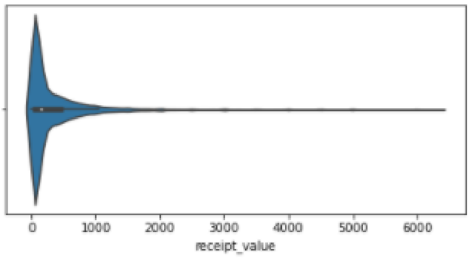
In [35]:
```
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
```

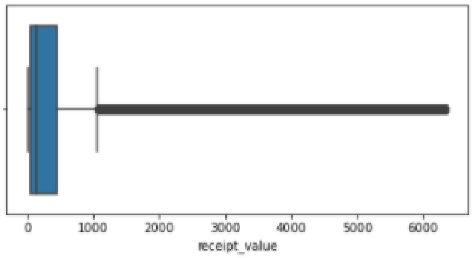
In [36]:
```
def get_outlier_boundaries(df, col, outlier_boundary_type: OutlierBoundaryType):
    if outlier_boundary_type == OutlierBoundaryType.SIGMA:
        K1 = 3
        lower_boundary = df[col].mean() - (K1 * df[col].std())
        upper_boundary = df[col].mean() + (K1 * df[col].std())

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

    return lower_boundary, upper_boundary
```

Вычисление верхней и нижней границы

```
In [37]: for col in x_listx:
             for obt in OutlierBoundaryType:
                 # Calculating the upper and lower bounds
                 lower_boundary, upper_boundary = get_outlier_boundaries(dataset2, col, obt)
                 # Flags for removing outliers
                 outliers_temp = np.where(dataset2[col] > upper_boundary, True,
                                          np.where(dataset2[col] < lower_boundary, True, False
                 # Deleting data based on the flag
                 dataset2_trimmed = dataset2.loc[~(outliers_temp), ]
                 title = 'field -{},method -{},rows-{}'.format(col, obt, dataset2_trimmed.shape
                 diagnostic_plots(dataset2_trimmed, col, title)
```

field -receipt_value,method -OutlierBoundaryType.SIGMA,rows-2975608

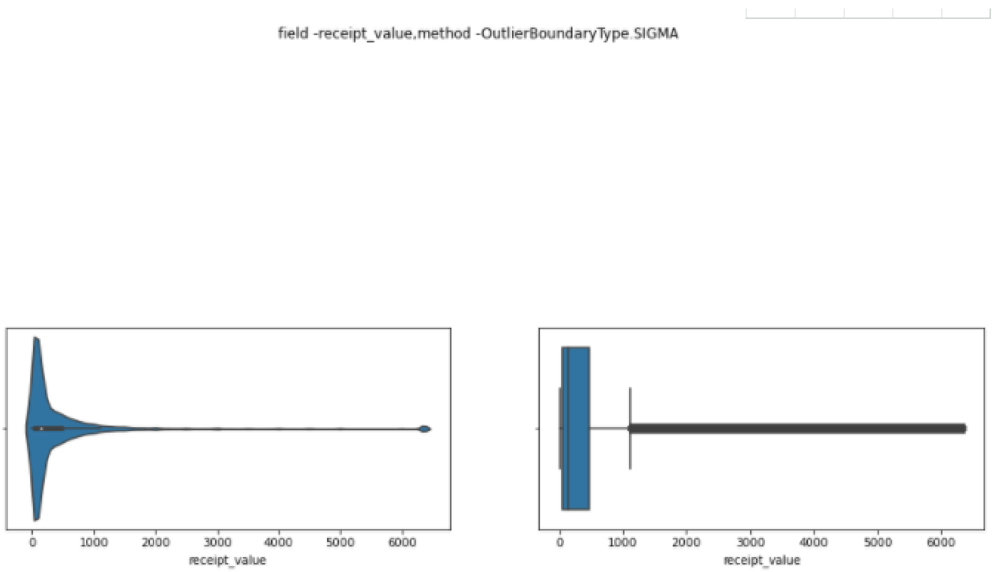




Замена выбросов

Вычисление верхней и нижней границы

In [38]:
```
for col in x_listx:
    for obt in OutlierBoundaryType:
        # Calculating the upper and lower bounds
        lower_boundary, upper_boundary = get_outlier_boundaries(dataset2, col, obt)
        # Data change
        dataset2[col] = np.where(dataset2[col] > upper_boundary, upper_boundary,
                                 np.where(dataset2[col] < lower_boundary, lower_bounda
        title = 'field -{},method -{}'.format(col, obt)
        diagnostic_plots(dataset2, col, title)
```



Обработка нестандартного признака

Сконвертируем дату и время в нужный формат

In [39]:
```
# Convert the date and time to the desired format
dataset2['dt'] = dataset2.apply(lambda x: pd.to_datetime(x['receipt_date'], format='%
```

In [40]:
```
dataset2.head()
```

Out[40]:

_name	receipt_social_security_number	receipt_description	establishment_name	receipt_value	dt
ɔelardo ıarinha	3.530749e+12	Fuels and lubricants.	AUTO POSTO 314 NORTE LTDA	70.0	2013-03-27
ɔelardo ıarinha	8.202116e+12	Fuels and lubricants.	AUTO POSTO AEROPORTO LTDA	104.0	2013-07-24
ɔelardo ıarinha	8.202116e+12	Fuels and lubricants.	AUTO POSTO AEROPORTO LTDA	100.0	2013-02-17
ɔelardo ıarinha	8.202116e+12	Fuels and lubricants.	AUTO POSTO AEROPORTO LTDA	100.0	2013-03-15
ɔelardo ıarinha	8.202116e+12	Fuels and lubricants.	AUTO POSTO AEROPORTO LTDA	77.0	2013-01-27

In [41]:
```
dataset2.dtypes
```

Out[41]:
```
bugged_date                              int64
receipt_date                            object
deputy_id                                int64
political_party                         object
state_code                              object
deputy_name                             object
receipt_social_security_number         float64
receipt_description                     object
establishment_name                      object
receipt_value                          float64
dt                              datetime64[ns]
dtype: object
```

In [42]:
```
# year
dataset2['year'] = dataset2['dt'].dt.year
# month
dataset2['month'] = dataset2['dt'].dt.month
# day
dataset2['day'] = dataset2['dt'].dt.day
#week
dataset2['week'] = dataset2['dt'].dt.isocalendar().week
#day of week
dataset2['dayofweek'] = dataset2['dt'].dt.dayofweek
#day_name
dataset2['day_name'] = dataset2['dt'].dt.day_name()
dataset2['is_holiday'] = dataset2.apply(lambda x: 1 if x['dt'].dayofweek in [5,6] el
```

In [43]:
```
dataset2.head()
```

Out[43]:

tablishment_name	receipt_value	dt	year	month	day	week	dayofweek	day_name	is_holiday
AUTO POSTO 314 NORTE LTDA	70.0	2013-03-27	2013	3	27	13	2	Wednesday	0
AUTO POSTO \EROPORTO LTDA	104.0	2013-07-24	2013	7	24	30	2	Wednesday	0
AUTO POSTO \EROPORTO LTDA	100.0	2013-02-17	2013	2	17	7	6	Sunday	1
AUTO POSTO \EROPORTO LTDA	100.0	2013-03-15	2013	3	15	11	4	Friday	0
AUTO POSTO \EROPORTO LTDA	77.0	2013-01-27	2013	1	27	4	6	Sunday	1

Разница между датами

In [44]:
```
# Difference between dates
dataset2['now'] = datetime.datetime.today()
dataset2['diff'] = dataset2['now'] - dataset2['dt']
dataset2.dtypes
```

Out[44]:
```
bugged_date                             int64
receipt_date                           object
deputy_id                               int64
political_party                        object
state_code                             object
deputy_name                            object
receipt_social_security_number        float64
receipt_description                    object
establishment_name                     object
receipt_value                         float64
dt                             datetime64[ns]
year                                    int64
month                                   int64
day                                     int64
week                                   UInt32
dayofweek                               int64
day_name                               object
is_holiday                              int64
now                            datetime64[ns]
diff                          timedelta64[ns]
dtype: object
```

In [45]:
```
dataset2.head()
```

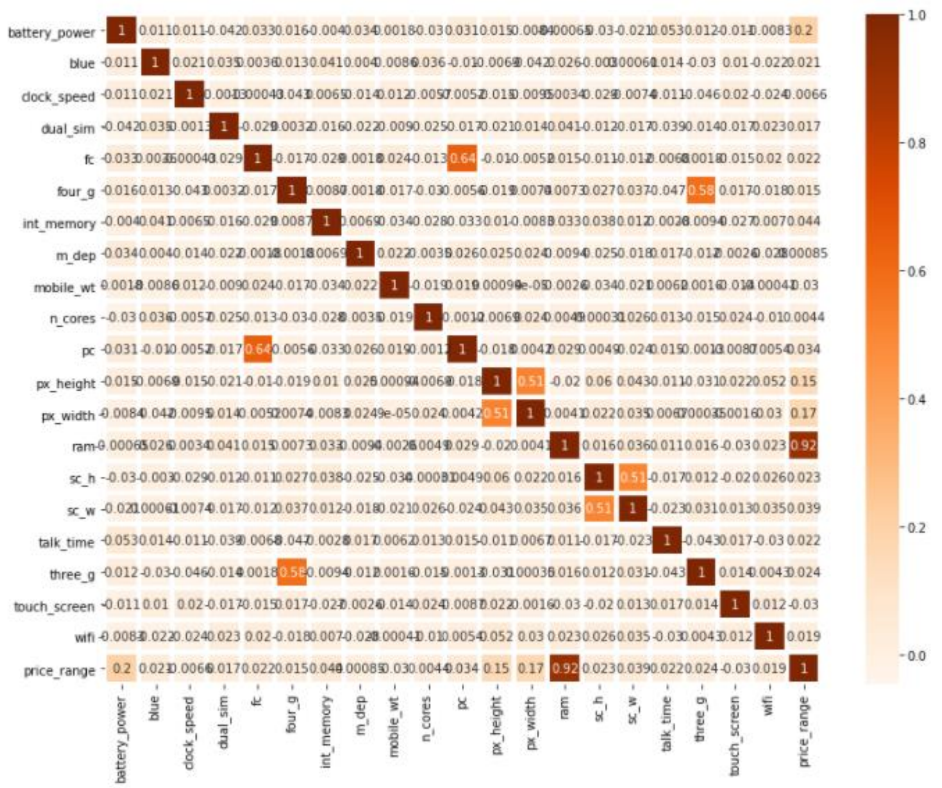
Out[45]:

dt	year	month	day	week	dayofweek	day_name	is_holiday	now	diff
2013-03-27	2013	3	27	13	2	Wednesday	0	2021-04-24 22:32:37.828917	2950 days 22:32:37.828917
2013-07-24	2013	7	24	30	2	Wednesday	0	2021-04-24 22:32:37.828917	2831 days 22:32:37.828917
2013-02-17	2013	2	17	7	6	Sunday	1	2021-04-24 22:32:37.828917	2988 days 22:32:37.828917
2013-03-15	2013	3	15	11	4	Friday	0	2021-04-24 22:32:37.828917	2962 days 22:32:37.828917
2013-01-27	2013	1	27	4	6	Sunday	1	2021-04-24 22:32:37.828917	3009 days 22:32:37.828917

Отбор признаков из группы методов фильтрации (корреляция признаков

```
In [46]: plt.figure(figsize=(13,10))
         sns.heatmap(dataset.corr(), cmap="Oranges", annot=True, linewidths=3)
```

```
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x25a67ebba90>
```



Формирование DataFrame с сильными корреляциями

```
In [47]: # Forming a DataFrame with strong correlations# Ф о р м и р о в а н и е  DataFrame  с
         def make_corr_df(df):
             cr = dataset.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 [48]: make_corr_df(dataset)
```

Out[48]:

	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

Обнаружение групп коррелирующих признаков

In [49]:
```
# Finding groups of correlating features
def corr_groups(cr):
    grouped_feature_list = []
    correlated_groups = []

    for feature in cr['f1'].unique():
        if feature not in grouped_feature_list:
            # finding correlating features
            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

In [50]:
```
# Groups of correlating features
corr_groups(make_corr_df(dataset))
```

Out[50]:
```
[['ram', 'price_range'], ['pc', 'fc']]
```

Отбор признаков из группы методов обертывания (алгоритм полного перебора)

In [51]:
```
from sklearn.neighbors import KNeighborsClassifier
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS

knn = KNeighborsClassifier(n_neighbors=3)
```

In [53]:
```
efs1 = EFS(knn,
           min_features=2,
           max_features=4,
           scoring='accuracy',
           print_progress=True,
           cv=5)

efs1 = efs1.fit(X_train, y_train, custom_feature_names=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: 0.91
Best subset (indices): (0, 11, 12, 13)
Best subset (corresponding names): ('battery_power', 'px_height', 'px_width', 'ram')
```

Отбор признаков из группы методов вложения (логистическая регрессия)

In [54]:
```
from sklearn.linear_model import LogisticRegression
# Using L1-regularization
e_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1', max_iter=500, ra
e_lr1.fit(X_train, y_train)
# Regression coefficients
e_lr1.coef_
```

Out[54]:
```
array([[-2.70452441e-02,  4.86917289e-01,  7.53960997e-02,
         2.42655394e-01, -1.03792618e-02,  4.48882816e-01,
        -3.43323356e-02, -1.79216643e+00,  5.94369224e-02,
        -1.87823726e-01,  8.63291606e-03, -1.65084207e-02,
        -1.55837848e-02, -4.48210315e-02,  8.34519025e-03,
         9.88593343e-02,  6.99751440e-02,  1.03969109e-01,
         8.04880528e-01,  9.27226410e-01],
       [-2.98527424e-04,  4.53473074e-03, -9.72402029e-02,
         2.28735020e-01,  4.22627166e-04,  2.15533944e-02,
         1.70099617e-03,  1.89967990e-01, -1.20477360e-06,
        -7.22136052e-02, -7.16034483e-04,  5.42037827e-05,
        -3.72738803e-05, -5.07464099e-04, -2.85761207e-03,
        -2.04429418e-02, -5.94962702e-04,  2.37288714e-02,
         8.50541290e-02, -7.33459968e-02],
       [-5.52497775e-05, -7.47980055e-02, -9.27772244e-02,
        -1.96632759e-01,  1.00512531e-02, -2.39809101e-01,
        -2.49915488e-03, -1.72446362e-01,  4.97713019e-03,
         4.35743588e-02, -4.46586211e-03,  2.10249923e-04,
        -9.00931806e-06,  5.20940619e-04, -3.19549938e-02,
         1.17428689e-02,  2.84161857e-03,  1.63129914e-01,
```

In [55]:
```
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[55]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True])
```

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Обработка признаков (часть2)» [Электронный ресурс] https://github.com/ugapanyuk/ml_course_2021/wiki/LAB_MMO__FEATURES