МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И.УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №1
по дисциплине «Машинное обучение»
Тема: Предобработка данных

Студент гр. 6304 _____ Корытов П.В.
Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

1. Цель работы

Ознакомиться с методами предобработки данных из библиотеки *Scikit Learn*.

2. Ход работы

2.1. Загрузка данных

1. Загружен указанный набор данных, проведены преобразования данных (рис. 1)

	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium
0	75.0	582	20	265000.00	1.9	130
1	55.0	7861	38	263358.03	1.1	136
2	65.0	146	20	162000.00	1.3	129
3	50.0	111	20	210000.00	1.9	137
4	65.0	160	20	327000.00	2.7	116
...	...	...	...	...	...	...
294	62.0	61	38	155000.00	1.1	143
295	55.0	1820	38	270000.00	1.2	139
296	45.0	2060	60	742000.00	0.8	138
297	45.0	2413	38	140000.00	1.4	140
298	50.0	196	45	395000.00	1.6	136

Рисунок 1 – Вывод набора данных

2. Построены гистограммы для выделенных признаков (рис. 2)

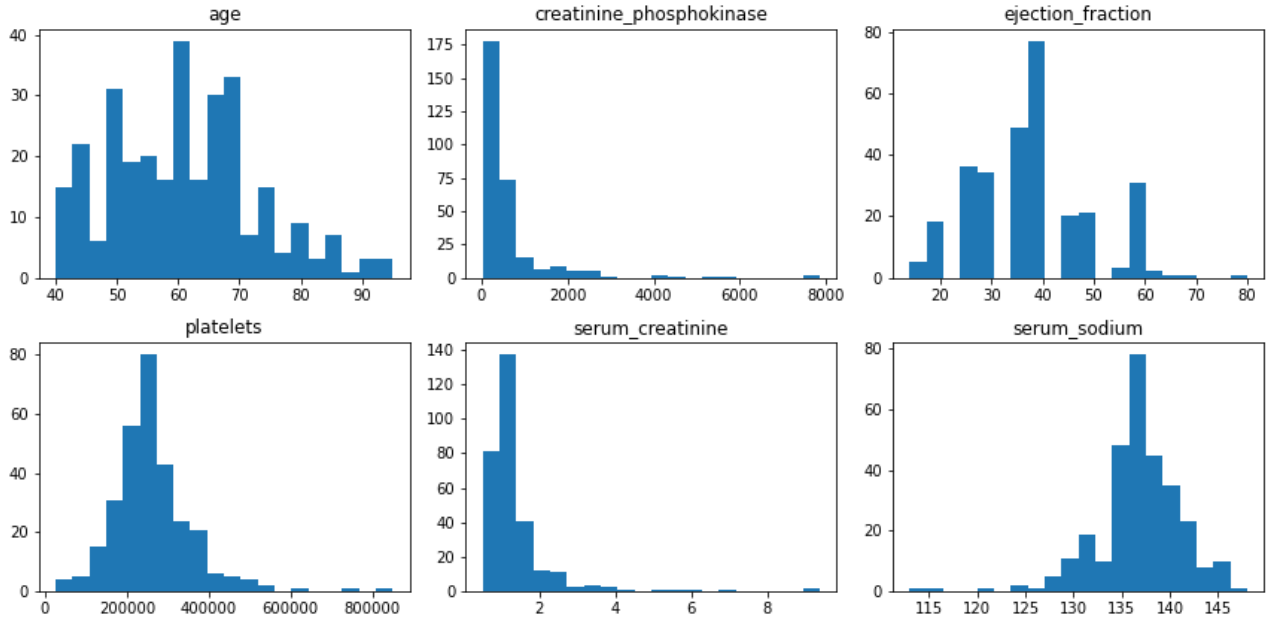

Рисунок 2 – Гистограммы призаков

2.2. Стандартизация данных

1. Проведена нормализация данных с помощью `StandardScaler` на основе первых 150 наблюдений. Гистограммы стандартизованных данных представлены на рис. 3.

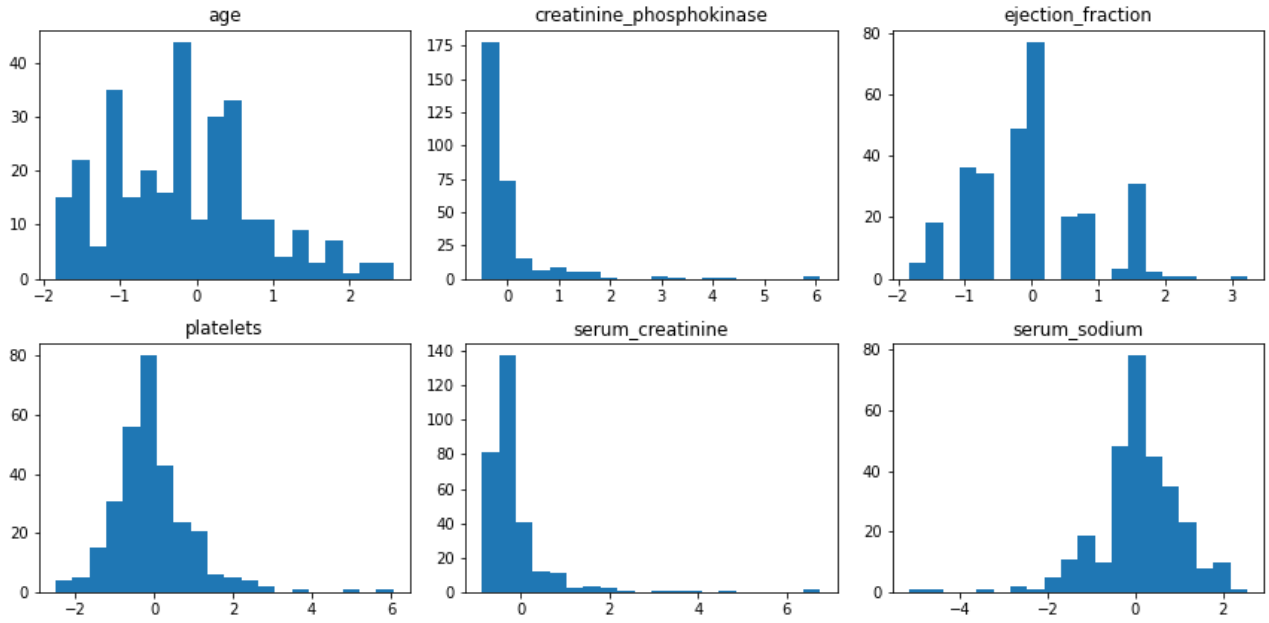

Рисунок 3 – Гистограммы нормализованных признаков (на осн. первых 150)

2. Проведена та же процедура на всем датасете. Результаты на рис. 4

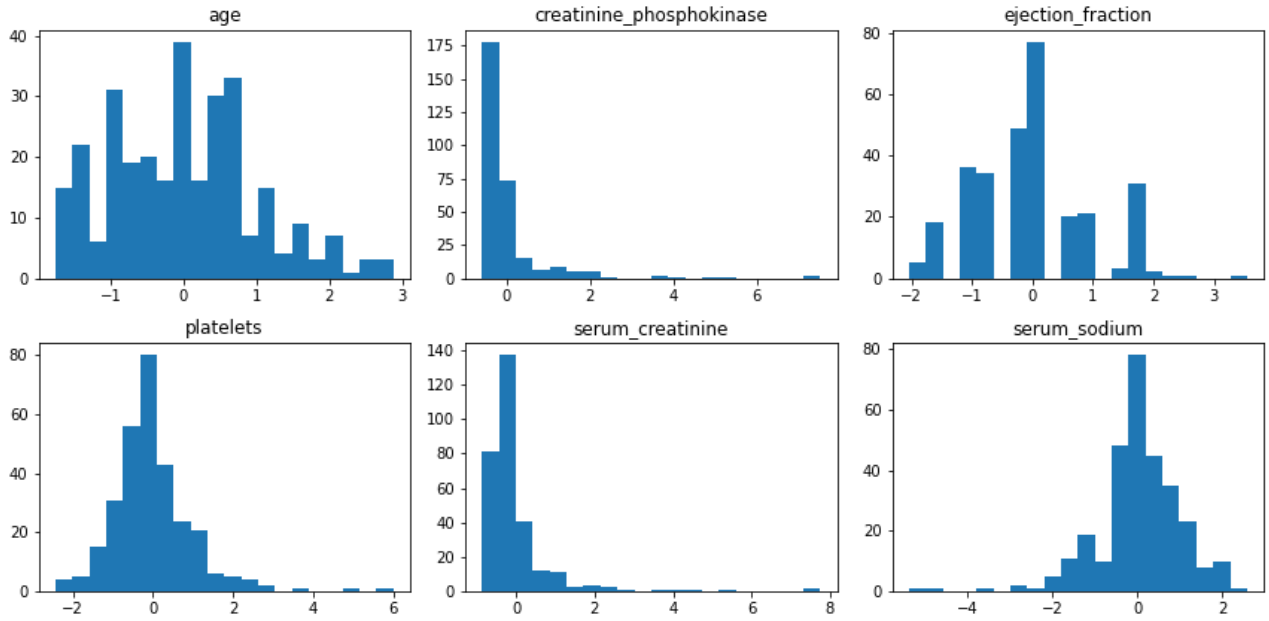

Рисунок 4 – Гистограммы нормализованных признаков

4. Вычислено мат. ожидание и СКО для исходных данных и обоих порций нормализованных данных. Результаты в таблице 1.

Таблица 1. Признаки

Признак	age	creatinine...	ejection_f...	platelets	serum_crea...	serum_sodi...
Среднее (исх.)	60.8339	581.8395	38.0836	263358.0293	1.3939	136.6254
Среднее (стандарт. 150)	-0.1697	-0.0213	0.0105	-0.0352	-0.1086	0.0379
Среднее (стандарт. 150 scaler)	62.9467	607.1533	37.9467	266746.7495	1.5206	136.4533
Среднее (стандарт. полн.)	0.0000	0.0000	-0.0000	0.0000	0.0000	-0.0000
Среднее (стандарт. полн. scaler)	60.8339	581.8395	38.0836	263358.0293	1.3939	136.6254
СКО (исх)	11.8749	968.6640	11.8150	97640.5477	1.0328	4.4051
СКО (стандарт. 150)	0.9538	0.8142	0.9061	1.0151	0.8854	0.9704
СКО (стандарт. 150 scaler)	12.4498	1189.7432	13.0393	96191.7902	1.1664	4.5396
СКО (стандарт. полн.)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
СКО (стандарт. полн. scaler)	11.8749	968.6640	11.8150	97640.5477	1.0328	4.4051

Судя по результатам в таблице 1 и рис. 4, `StandardScaler` центрирует данные относительно среднего и масштабирует относительно дисперсии. Предположительная формула:

$$Y_i = \frac{X_i - M[X]}{\sqrt{D[X]}}, \tag{2.1}$$

где X — исходные данные, Y — преобразованные данные.

В объекте scaler записывается дисперсия и мат. ожидание исходных данных. При ограничении размера выборки для настройки результаты нормализуются менее качественно, т.е. мат. ожидание и СКО отличаются от 0 и 1.

2.3. Приведение к диапазону

1. Данные приведены к диапазону в помощью `MinMaxScaler`. Гистограммы приведены на рис. 5.

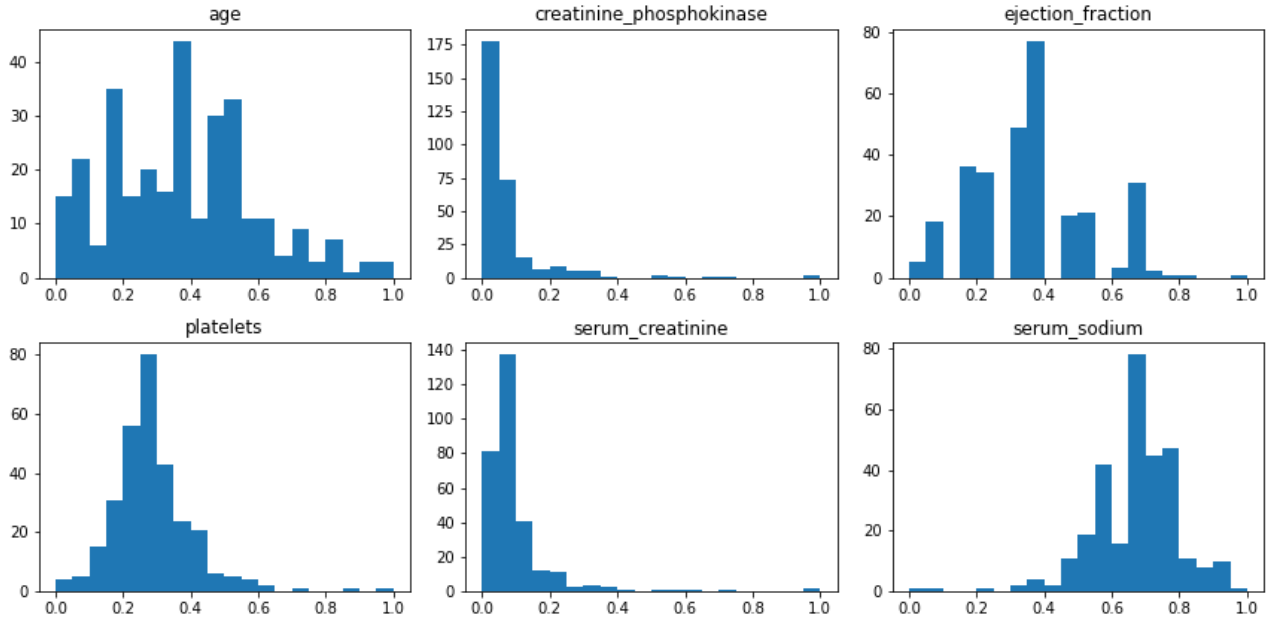

Рисунок 5 – Гистограммы данных после `MinMaxScaler`

Исходя из гистограмм, `MinMaxScaler` масштабирует данные к промежутку $[0, 1]$.

2. Из атрибутов scaler-а получены значения атрибутов. Результаты приведены в таблице 2.

Таблица 2. Минимальные и максимальные значения признаков

Признак	Минимум	Максимум
age	40	95
creatinine_phosphokinase	23	7861
ejection_fraction	14	80
platelets	25100	850000
serum_creatinine	0.5	9.4
serum_sodium	113	148

3. Проведено приведение с помощью `MaxAbsScaler` и `RobustScaler`. Гистограммы данных приведены на рис. 6 и рис. 7.

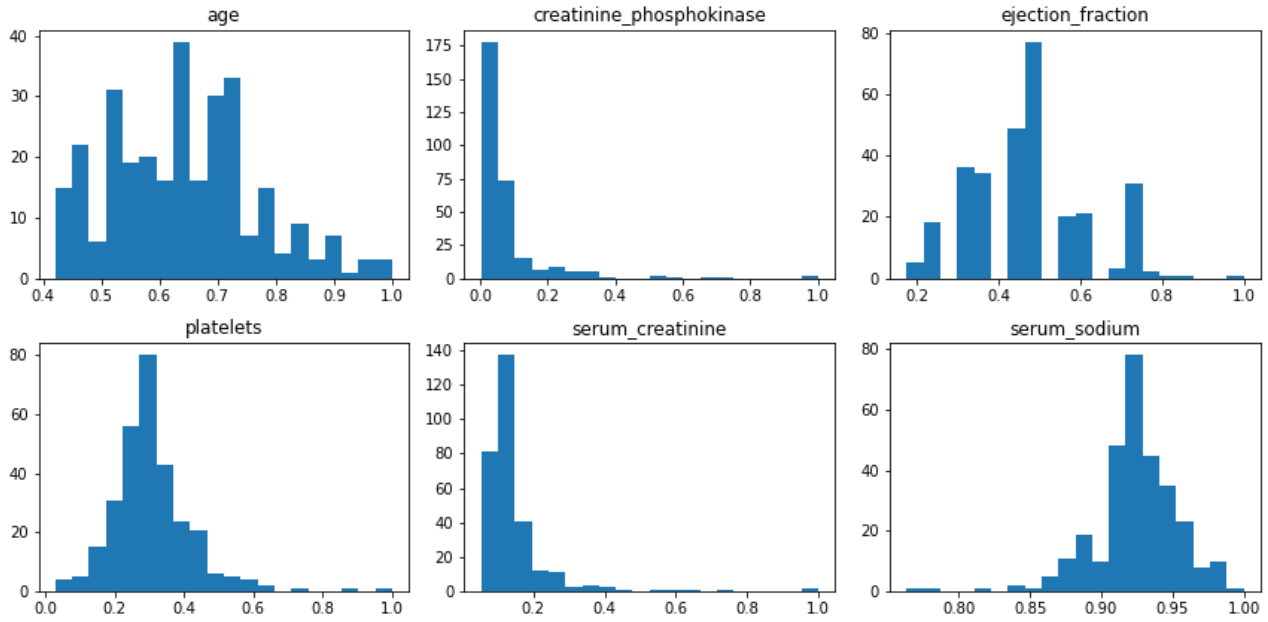

Рисунок 6 – Гистограммы данных после `MaxAbsScaler`

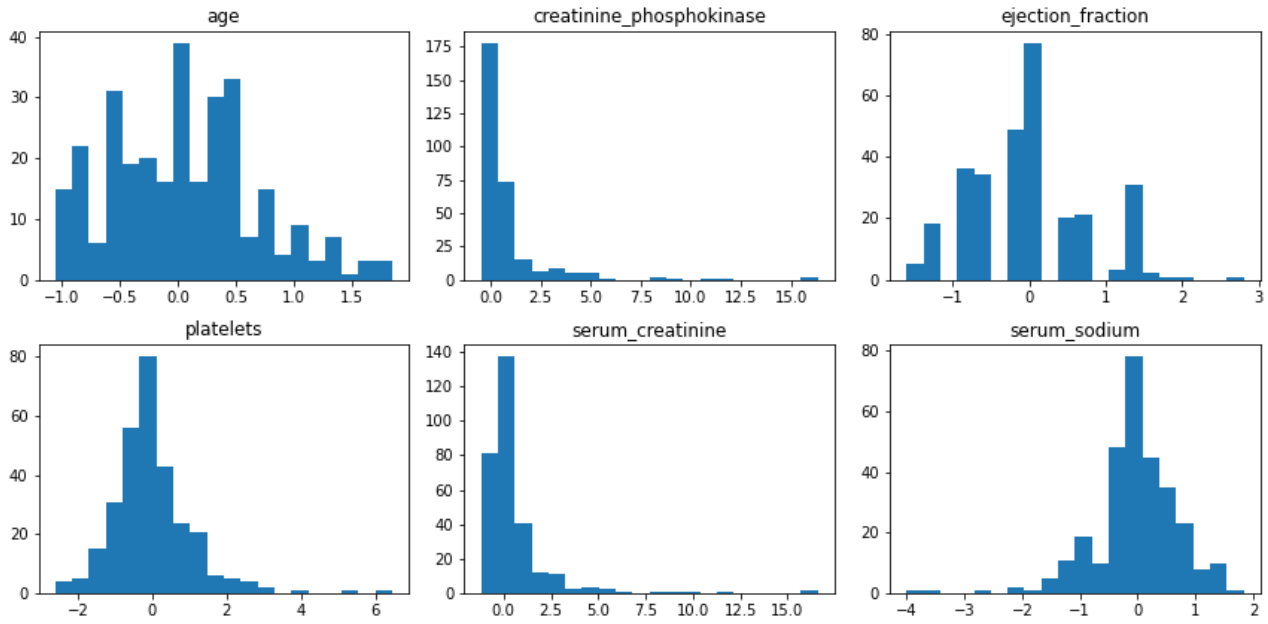

Рисунок 7 – Гистограммы данных после `RobustScaler`

`MaxAbsScaler` изменяет данные таким образом, чтобы максимальное значение по модулю было равно 1. `RobustScaler` центрирует по медиане и масштабирует данные относительно диапазона между 25-м и 75-м процентилем.

4. Написана функция для приведения данных к диапазону $[-5, 10]$. Результат приведен в листинге 1.

Листинг 1. Функция для приведения данных к диапазону $[-5, 10]$

```
1  def fit_5_10(data):
2      data = data.copy()
3      for col in range(data.shape[1]):
4          min_, max_ = np.min(data[:, col]), np.max(data[:, col])
5          data[:, col] = [(x - min_) / (max_ - min_) * 15 - 5 for x
           ↪  in data[:, col]]
6      return data
```

Гистограммы для этого преобразования представлены на рис. 8.

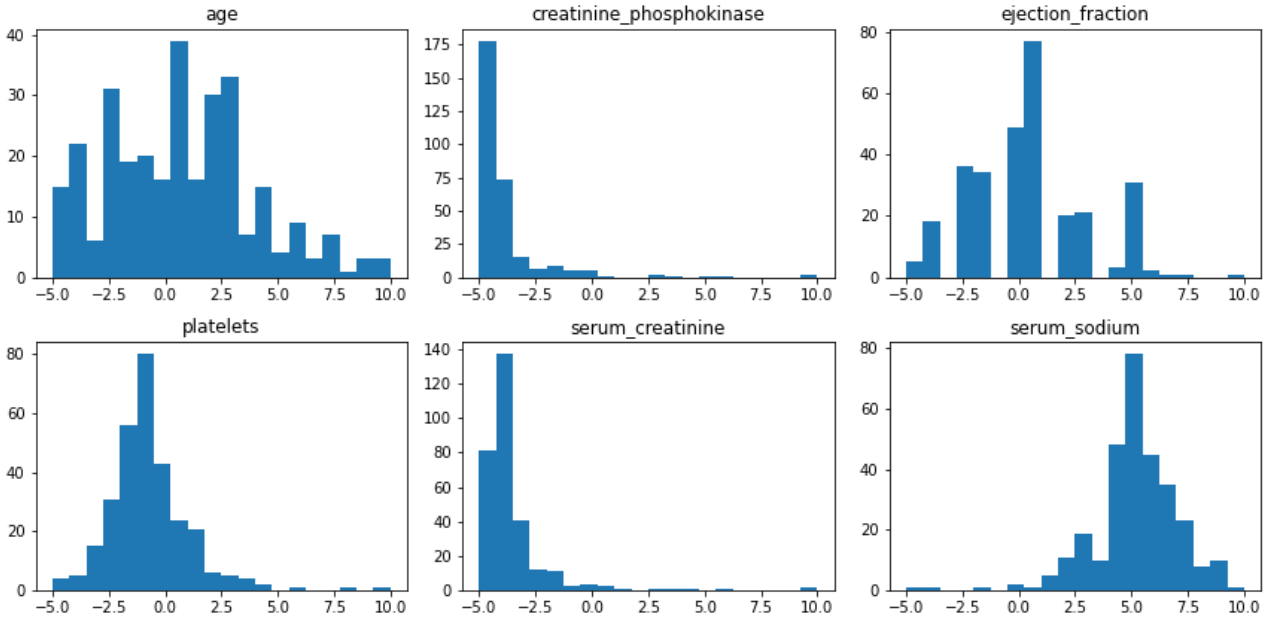

Рисунок 8 – Гистограммы после `fit_5_10`

2.4. Нелинейные преобразования

1. С помощью `QuantileTransform` данные приведены к равномерному и нормальному распределению. Результаты представлены на рис. 9, 10.

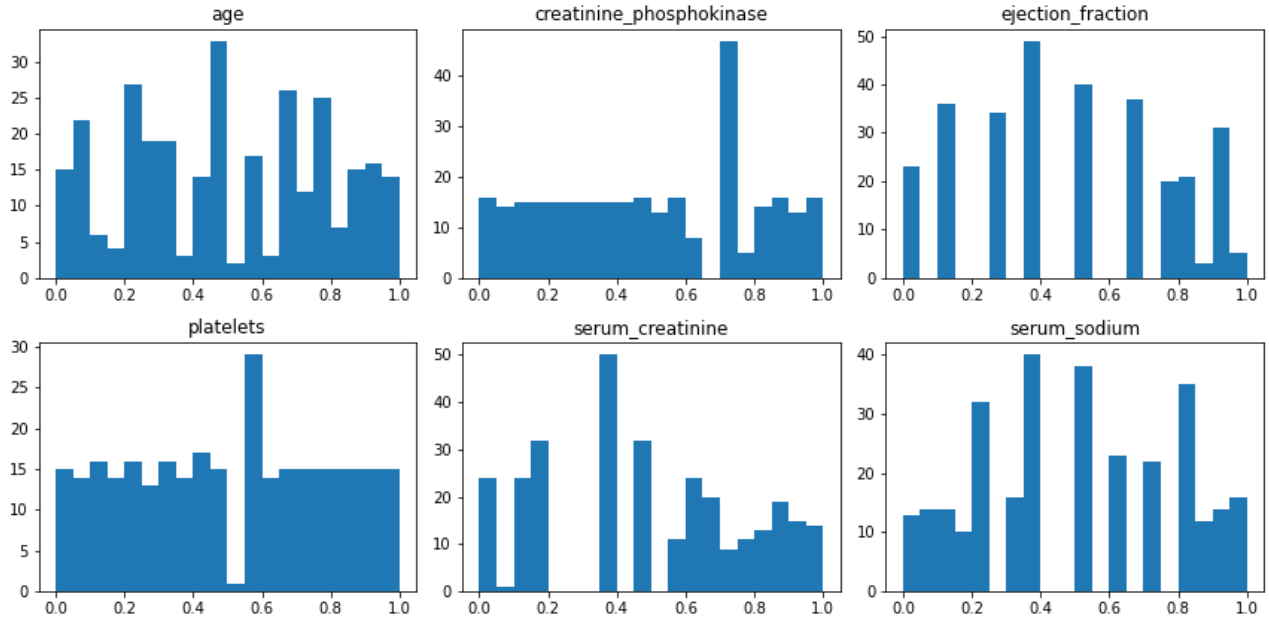

Рисунок 9 – Гистограммы после `QuantileTransform` с равномерным распределением

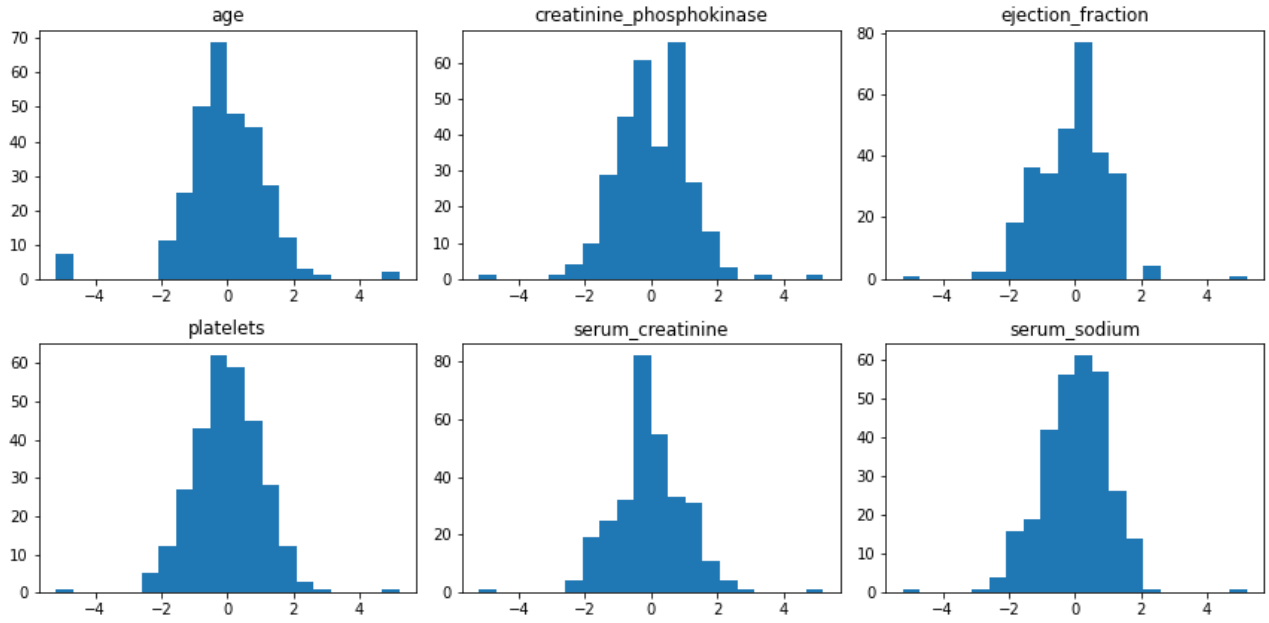

Рисунок 10 – Гистограммы после `QuantileTransform` с нормальным распределением

Параметр `n_quantiles` определяет количество вычисляемых процентилей в ходе настройки. Увеличение повышает частоту дискретизации функции распределения.

2. Данные приведены к нормальному распределению через `PowerTransformer`. Результаты на рис. 11.

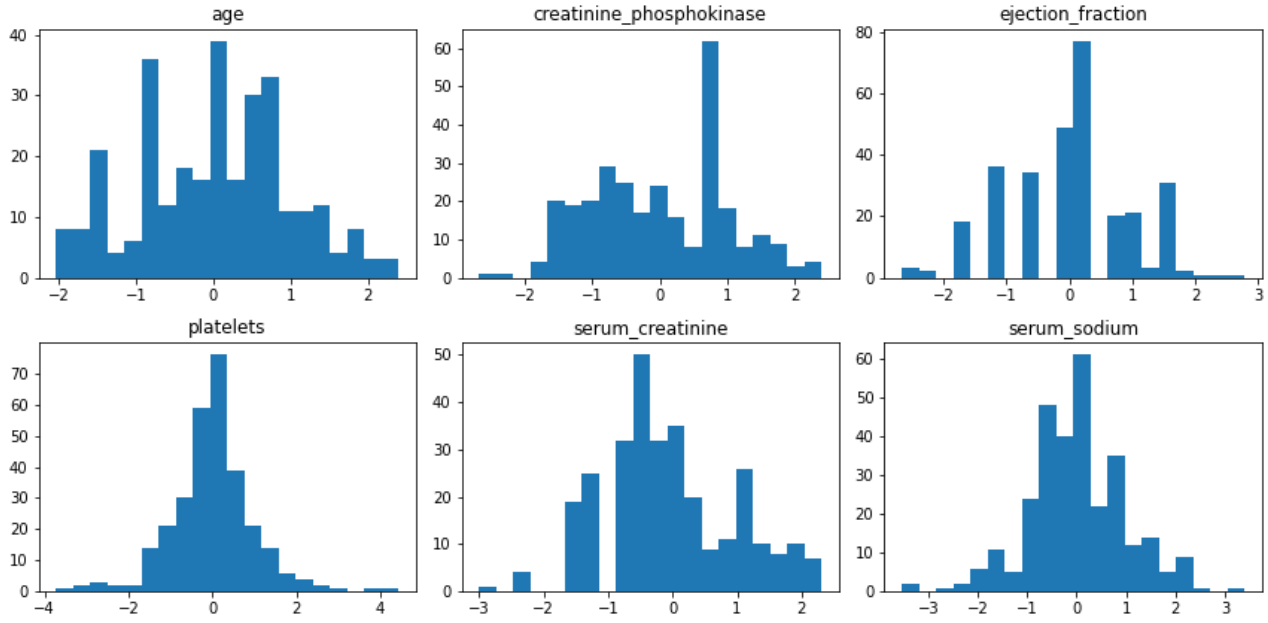

Рисунок 11 – Гистограммы после `PowerTransformer`

2.5. Дискретизация признаков

1. Проведена дискретизация признаков с помощью `KBinsDiscretizer`. Гистограммы дискретизованных данных приведены на рис 12.

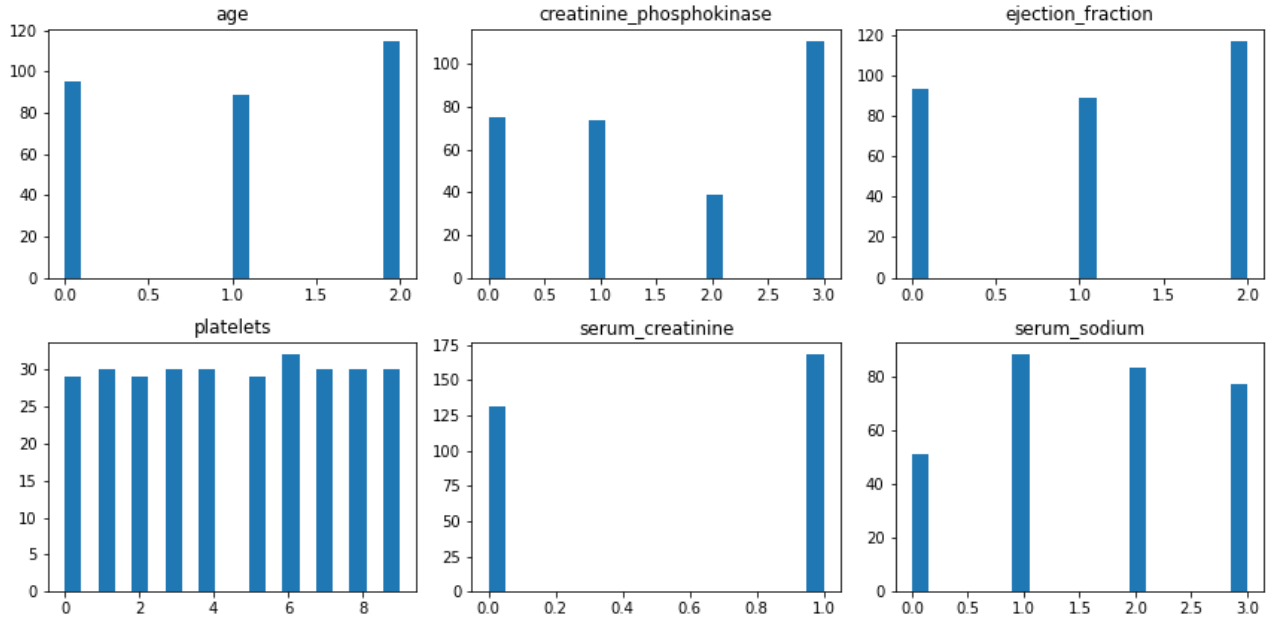

Рисунок 12 – Гистограммы после `KBinsDiscretizer`

Поскольку значения по оси ординат являются числовыми идентификаторами дискретных значений, построение гистограммы не имеет смысла.

3. Выводы

Произведено знакомство с методами предобработки данных библиотеки *Scikit Learn*.

Проведена стандартизация данных; установлено, что стандартизация с учетом неполного набора данных снижает качество выходных данных.

Проведено приведение данных к диапазону. Гистограммы данных, приведенных к диапазону, схожи с гистограммами стандартизованных данных.

Также проведены нелинейные преобразования данных. Предположительно, использование `QuantileTransform` может иметь смысл при наличии в данных выбросов; однако в этом случае искажается структура данных.

Также проведена дискретизация данных.

ПРИЛОЖЕНИЕ А

Исходный код программы

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import pandas as pd

from tabulate import tabulate
from IPython.core.debugger import set_trace
from IPython.display import display
from matplotlib import pyplot as plt
from sklearn import preprocessing


# In[2]:


df =
 ↪  pd.read_csv('../data/heart_failure_clinical_records_dataset.csv')
df =
 ↪  df.drop(columns=['anaemia','diabetes','high_blood_pressure','sex','sm

display(df)


# In[3]:


fig, axes = plt.subplots(2, 3, figsize=(12, 6))
n_bins = 20

axes[0, 0].hist(df['age'].values, bins = n_bins)
axes[0, 0].set_title('age')

axes[0, 1].hist(df['creatinine_phosphokinase'].values, bins =
 ↪  n_bins)
axes[0, 1].set_title('creatinine_phosphokinase')

axes[0, 2].hist(df['ejection_fraction'].values, bins = n_bins)
axes[0, 2].set_title('ejection_fraction')

axes[1, 0].hist(df['platelets'].values, bins = n_bins)
axes[1, 0].set_title('platelets')
```

```
43
44  axes[1, 1].hist(df['serum_creatinine'].values, bins = n_bins)
45  axes[1, 1].set_title('serum_creatinine')
46
47  axes[1, 2].hist(df['serum_sodium'].values, bins = n_bins)
48  axes[1, 2].set_title('serum_sodium')
49
50  fig.tight_layout()
51
52  plt.savefig('./img/hist-1.png')
53  plt.show()
54
55
56  # In[4]:
57
58
59  data = df.to_numpy(dtype='float')
60
61
62  # In[5]:
63
64
65  scaler = preprocessing.StandardScaler().fit(data[:150,:])
66  data_scaled = scaler.transform(data)
67
68
69  # In[6]:
70
71
72  TITLES = ['age', 'creatinine_phosphokinase', 'ejection_fraction',
    ↪  'platelets', 'serum_creatinine', 'serum_sodium']
73
74  def plot_data(data_scaled):
75      fig, axes = plt.subplots(2, 3, figsize=(12, 6))
76      ax_order = [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2)]
77
78      for i, ax_ind in enumerate(ax_order):
79          axes[ax_ind].hist(data_scaled[:,i], bins = n_bins)
80          axes[ax_ind].set_title(TITLES[i])
81
82      fig.tight_layout()
83      return fig
84
85  plot_data(data_scaled)
86  plt.savefig('./img/hist-2.png')
87  plt.show()
88
89
```

```
# In[7]:


def calc_metrics(data):
    mean = [np.mean(col) for col in data.T]
    std = [np.std(col) for col in data.T]
    return mean, std

calc_metrics(data)


# In[8]:


def shorten(s):
    if len(s) < 10:
        return s
    return s[:10] + '...'

mean_src, std_src = calc_metrics(data)
mean_sc, std_sc = calc_metrics(data_scaled)

scaler2 = preprocessing.StandardScaler()
data_scaled2 = scaler2.fit_transform(data)
mean_sc2, std_sc2 = calc_metrics(data_scaled2)

plot_data(data_scaled2)
plt.savefig('./img/hist-3.png')
plt.show()

header = ['Признак', *[shorten(t) for t in TITLES]]
table = [
    ['Среднее (исх.)', *mean_src],
    ['Среднее (стандарт. 150)', *mean_sc],
    ['Среднее (стандарт. 150 scaler)', *scaler.mean_],
    ['Среднее (стандарт. полн.)', *mean_sc2],
    ['Среднее (стандарт. полн. scaler)', *scaler2.mean_],
    ['СКО (исх)', *std_src],
    ['СКО (стандарт. 150)', *std_sc],
    ['СКО (стандарт. 150 scaler)', *[np.sqrt(v) for v in
    ↪  scaler.var_]],
    ['СКО (стандарт. полн.)', *std_sc2],
    ['СКО (стандарт. полн. scaler)', *[np.sqrt(v) for v in
    ↪  scaler2.var_]]
]

latex_t1 = tabulate(table, headers=header,
↪  tablefmt='latex_booktabs', floatfmt=".4f")
```

```
135 with open('./output/t1.tex', 'w') as f:
136     f.write(latex_t1)
137
138
139 # In[9]:
140
141
142 min_max_scaler = preprocessing.MinMaxScaler()
143 min_max_data = min_max_scaler.fit_transform(data)
144 plot_data(min_max_data)
145
146 plt.savefig('./img/hist-min-max.png')
147 plt.show()
148
149
150 # In[10]:
151
152
153 header = ['Признак', 'Минимум', 'Максимум']
154 table = [
155     (title, min_, max_)
156     for title, min_, max_ in zip(TITLES, min_max_scaler.data_min_,
        ↪  min_max_scaler.data_max_)
157 ]
158
159 latex_t2 = tabulate(table, headers=header,
    ↪  tablefmt='latex_booktabs')
160 with open('./output/t2.tex', 'w') as f:
161     f.write(latex_t2)
162
163
164 # In[11]:
165
166
167 max_abs_data = preprocessing.MaxAbsScaler().fit_transform(data)
168 robust_data = preprocessing.RobustScaler().fit_transform(data)
169
170 plot_data(max_abs_data)
171 plt.savefig('./img/hist-max-abs.png')
172 plt.show()
173
174 plot_data(robust_data)
175 plt.savefig('./img/hist-robust.png')
176 plt.show()
177
178
179 # In[12]:
180
```

```
181
182 def fit_5_10(data):
183     data = data.copy()
184     for col in range(data.shape[1]):
185         min_, max_ = np.min(data[:, col]), np.max(data[:, col])
186         data[:, col] = [(x - min_) / (max_ - min_) * 15 - 5 for x in
        ↪  data[:, col]]
187     return data
188
189 data_5_10 = fit_5_10(data)
190 plot_data(data_5_10)
191 plt.savefig('./img/hist-5-10.png')
192 plt.show()
193
194
195 # In[13]:
196
197
198 quantile_transformer =
    ↪  preprocessing.QuantileTransformer(n_quantiles=100,
    ↪  random_state=0)
199 quantile_data = quantile_transformer.fit_transform(data)
200
201 plot_data(quantile_data)
202 plt.savefig('./img/hist-quantile.png')
203 plt.show()
204
205
206 # In[14]:
207
208
209 quantile_normal_transformer =
    ↪  preprocessing.QuantileTransformer(n_quantiles=100,
    ↪  random_state=0, output_distribution='normal')
210 quantile_normal_data =
    ↪  quantile_normal_transformer.fit_transform(data)
211
212 plot_data(quantile_normal_data)
213 plt.savefig('./img/hist-quantile-normal.png')
214 plt.show()
215
216
217 # In[15]:
218
219
220 power_transformer = preprocessing.PowerTransformer()
221 power_data = power_transformer.fit_transform(data)
222
```

```
223  plot_data(power_data)
224  plt.savefig('./img/hist-power.png')
225  plt.show()
226
227
228  # In[16]:
229
230
231  est = preprocessing.KBinsDiscretizer(n_bins=[3, 4, 3, 10, 2, 4],
     ↪  encode='ordinal')
232  disc_data = est.fit_transform(data)
233
234  plot_data(disc_data)
235  plt.savefig('./img/hist-disc.png')
236  plt.show()
237
238
239  # In[ ]:
```