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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Машинное обучение»

Тема: Предобработка данных

Студенты гр. 6304 _____ Григорьев И.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы

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

Ход работы

Загрузка данных

1. Датасет загружен в датафрейм, исключены бинарные признаки и признак времени.
2. Построены гистограммы признаков, которые приведены на рис. 1.

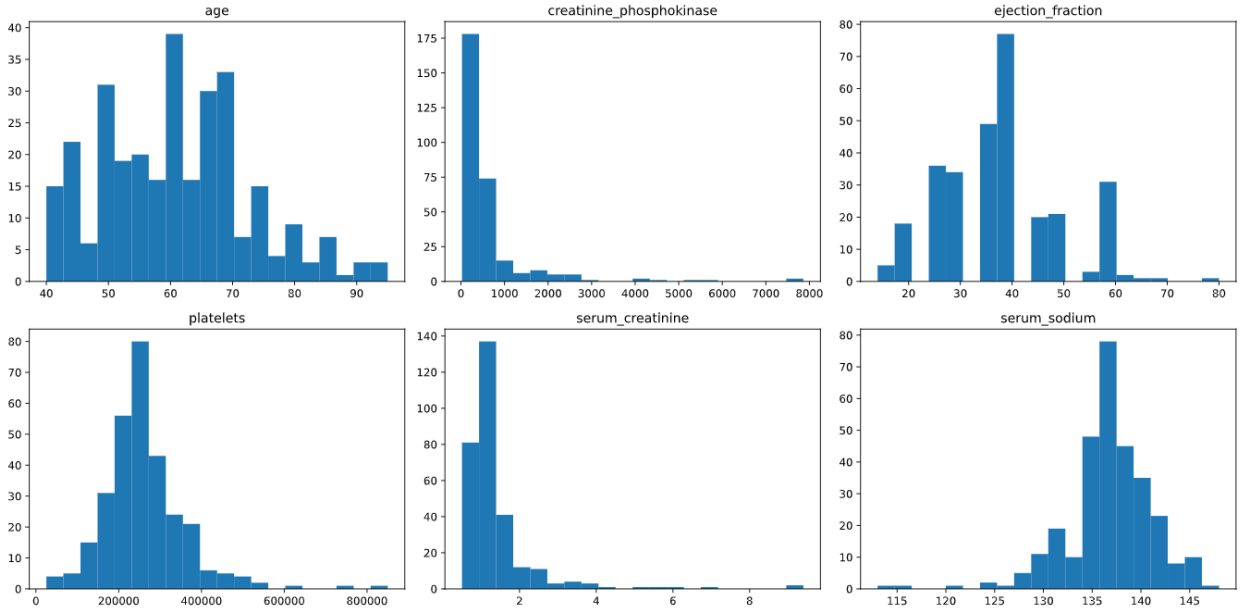

Рисунок 1 – Исходные данные

3. На основании гистограмм определены диапазоны значений для каждого из признаков, а также возле какого значения лежит наибольшее количество наблюдений (см. табл. 1).

Таблица 1

	Диапазон	Наибольшее количество наблюдений
age	(40, 95)	60
creatinine_phosphokinase	(0, 7900)	250
ejection_fraction	(10, 80)	38
platelets	(25 000, 850 000)	250 000
serum_creatinine	(0.5, 9)	1

serum_sodium	(115, 147)	136

Стандартизация данных

1. Данные стандартизированы на основе первых 150 наблюдений, гистограммы представлены на рис. 2.

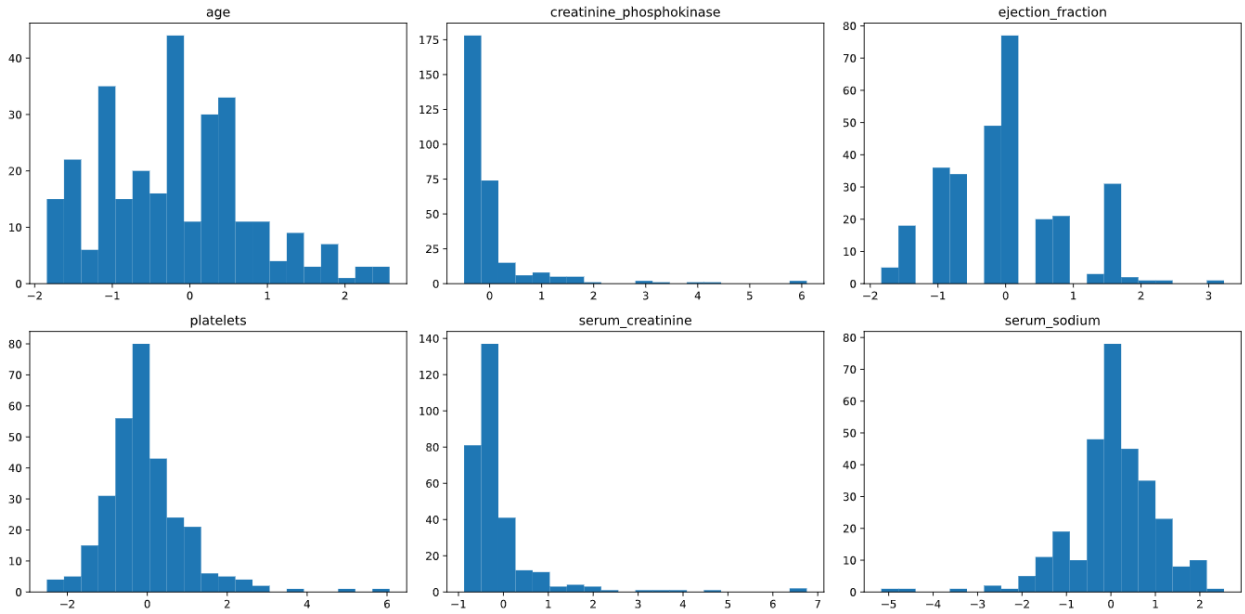

Рисунок 2 – Стандартизированные данные (на основе 150 первых наблюдений)

2. На основании гистограмм для стандартизированных данных определены диапазоны значений для каждого из признаков, а также возле какого значения лежит наибольшее количество наблюдений (см. табл. 2).

Таблица 2

	Диапазон	Наибольшее количество наблюдений
age	(-2, 2.5)	-0.15
creatinine_phosphokinase	(-0.5, 6)	-0.5
ejection_fraction	(-2, 3)	0
platelets	(-2.5, 6)	0
serum_creatinine	(-1, 7)	-0.5
serum_sodium	(-5, 2.5)	0

Из табл. 2 видно, что наибольшее количество наблюдений теперь располагается около нуля для всех признаков.

3. Данные стандартизированы на основе всех наблюдений. Рассчитаны математическое ожидание и СКО до стандартизации и после стандартизации (для первых 150 наблюдений и для всех наблюдений), результаты расчетов представлены в табл. 3.

Таблица 3

Данные		age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium
Исходные	mean	60.83	581.83	38.08	263358	1.39	136.62
	std	11.87	968.66	11.81	97640	1.03	4.41
Стандартизированные (150)	mean	-0.1697	-0.0213	0.0105	-0.0352	-0.1086	0.0379
	std	0.9538	0.8141	0.9061	1.0151	0.8854	0.9703
Стандартизированные	mean	0	0	0	0	0	0
	std	1	1	1	1	1	1

На основании результатов стандартизация имеет следующий вид:

$$Y = \frac{X - mean(X)}{std(X)}$$

В объекте *scaler mean_* и *var_* – это математическое ожидание и дисперсия набора данных, на основе которого будет произведена стандартизация.

Для неполной выборки (150 первых наблюдений) стандартизация выполняется менее качественно (математическое ожидание и СКО отличаются от 0 и 1 соответственно, хотя имеют близкие значения).

Приведение к диапазону

1. Данные приведены к диапазону с помощью *MinMaxScaler*, гистограммы представлены на рис. 2.

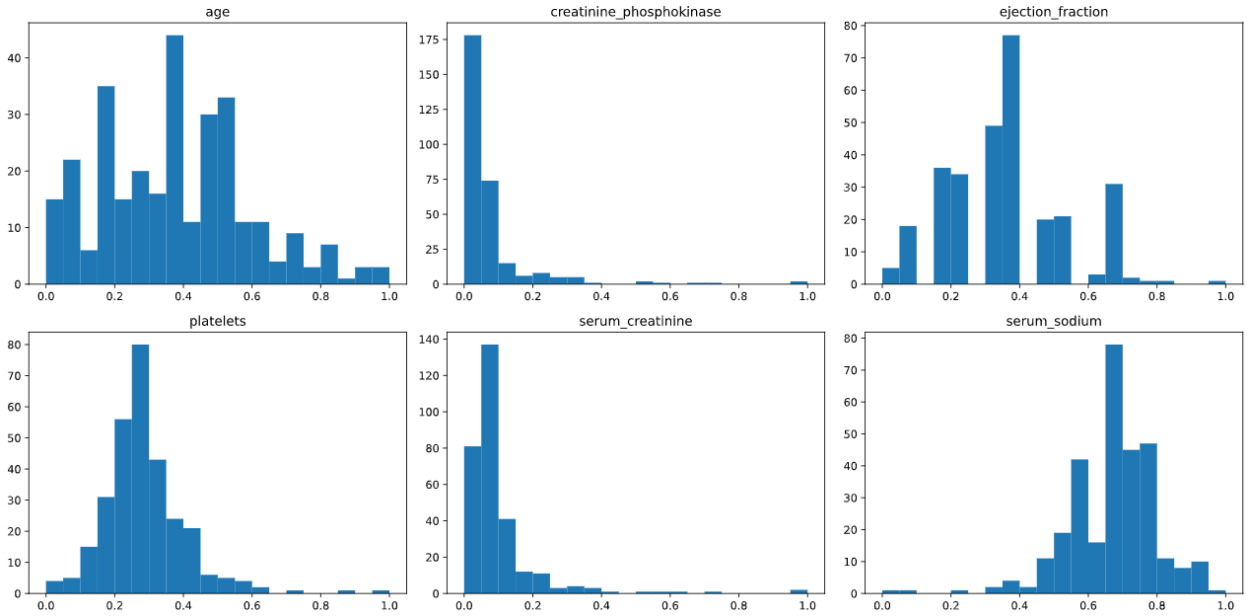

Рисунок 3 – Данные, приведенные к диапазону с помощью *MinMaxScaler*

Из рис. 3 видно, что данные приводятся к диапазону [0, 1].

Минимальные и максимальные значения данных для каждого признака из объекта *MinMaxScaler* приведены в таблице 4.

Таблица 4

	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium
min	4.00e+01	2.30e+01	1.40e+01	2.51e+04	5.00e-01	1.13e+02
max	9.500e+01	7.861e+03	8.000e+01	8.500e+05	9.400e+00	1.480e+02

2. Данные преобразованы с помощью *MaxAbsScaler* и *RobustScaler*, гистограммы представлены на рис. 4-5.

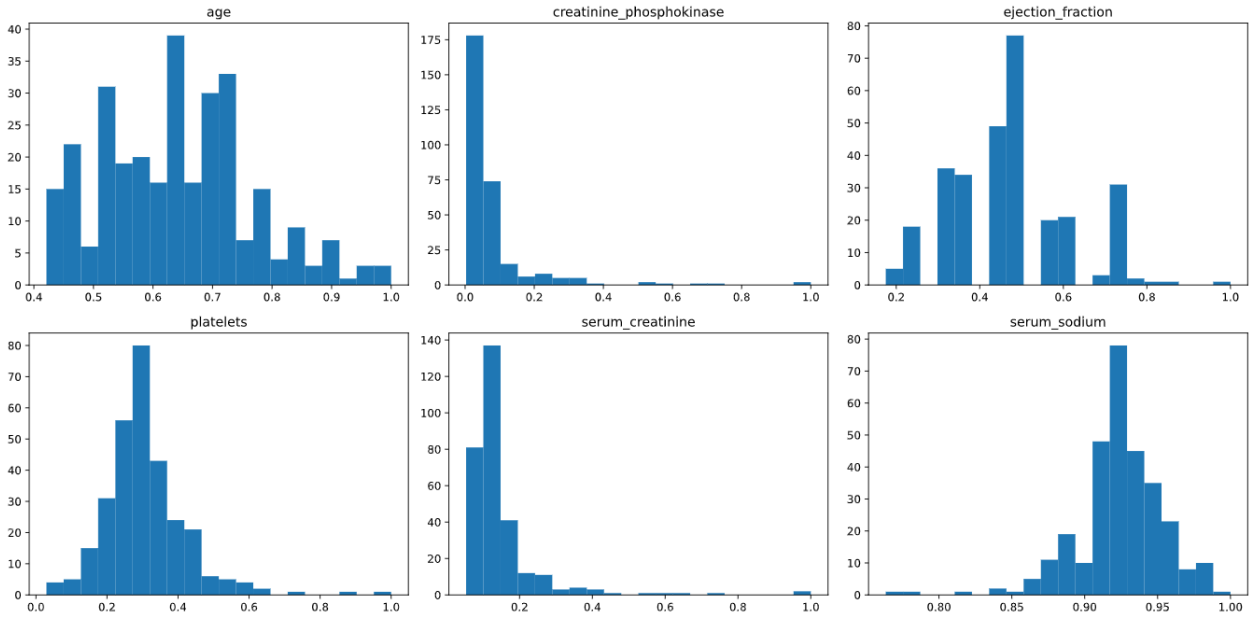

Рисунок 4 – Данные, преобразованные с помощью *MaxAbsScaler*

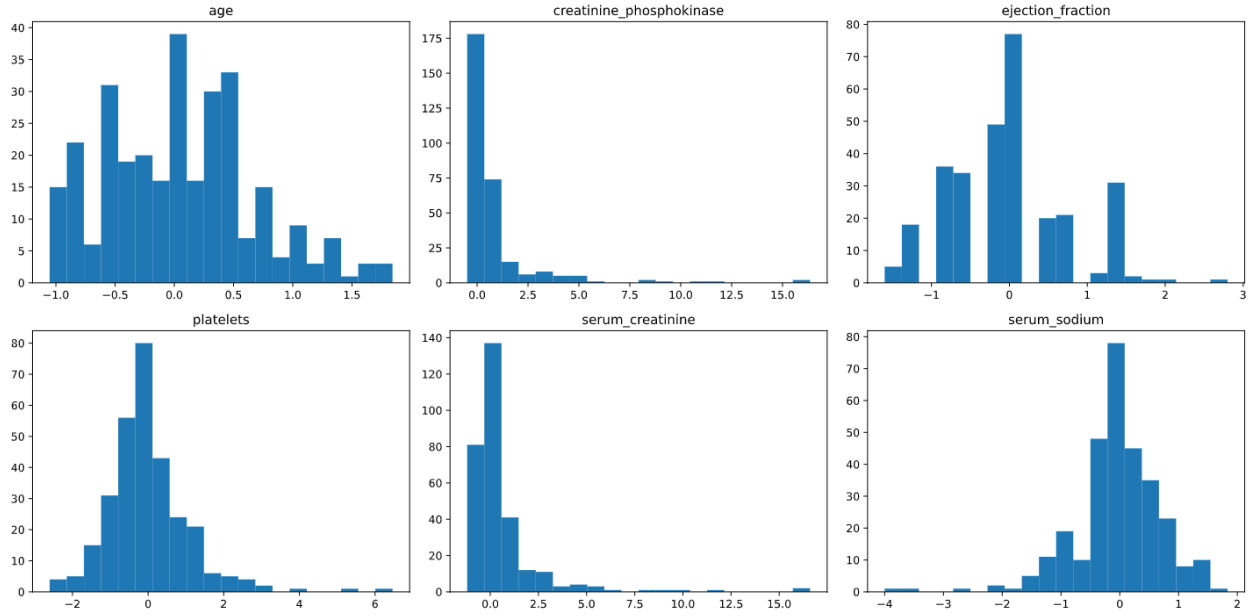

Рисунок 5 – Данные, преобразованные с помощью *RobustScaler*

MaxAbsScaler приводит данные к виду, когда максимальное значение по модулю равно 1. *RobustScaler* центрирует данные по медиане и масштабирует данные относительно диапазона между 1 и 3 квартилями.

3. Написана функция, которая приводит все данные к диапазону [-5, 10].

```
np.array([[((x - np.min(col)) / (np.max(col) - np.min(col))) * 15 - 5 for x in col] for col in data.T]).T
```

Данные, приведенные к диапазону [-5, 10], продемонстрированы на рис. 6.

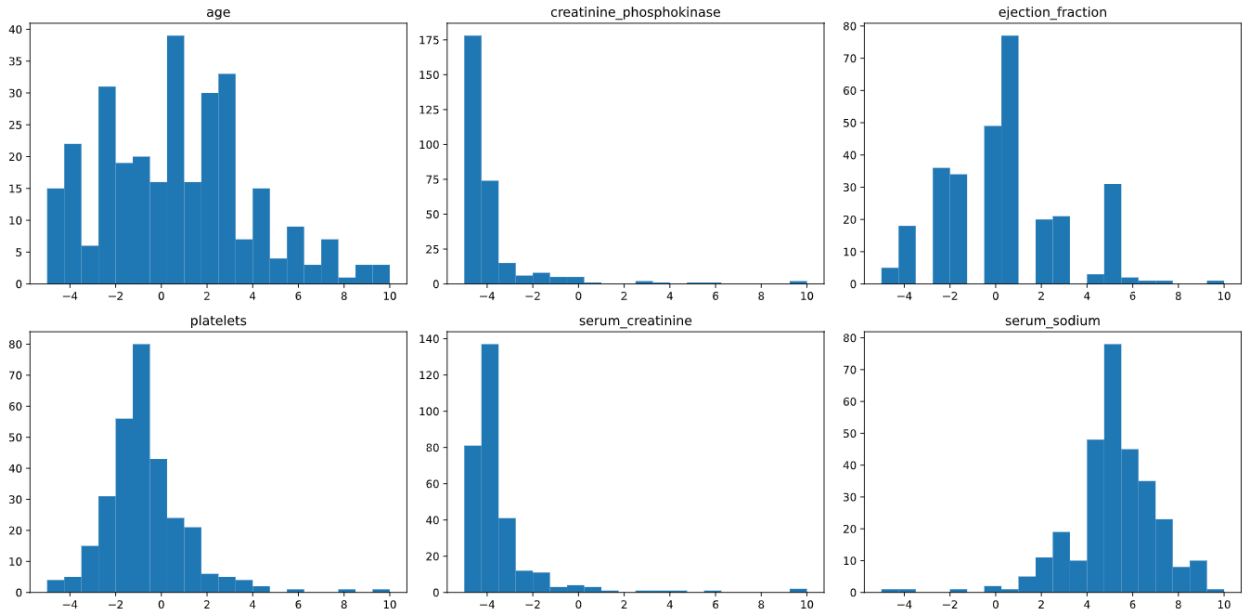

Рисунок 6 – Данные, приведенные к диапазону [-5, 10]

Нелинейные преобразования

1. С помощью *QuantileTransformer* данные приведены к равномерному и нормальному распределениям, гистограммы представлены на рис. 7-8.

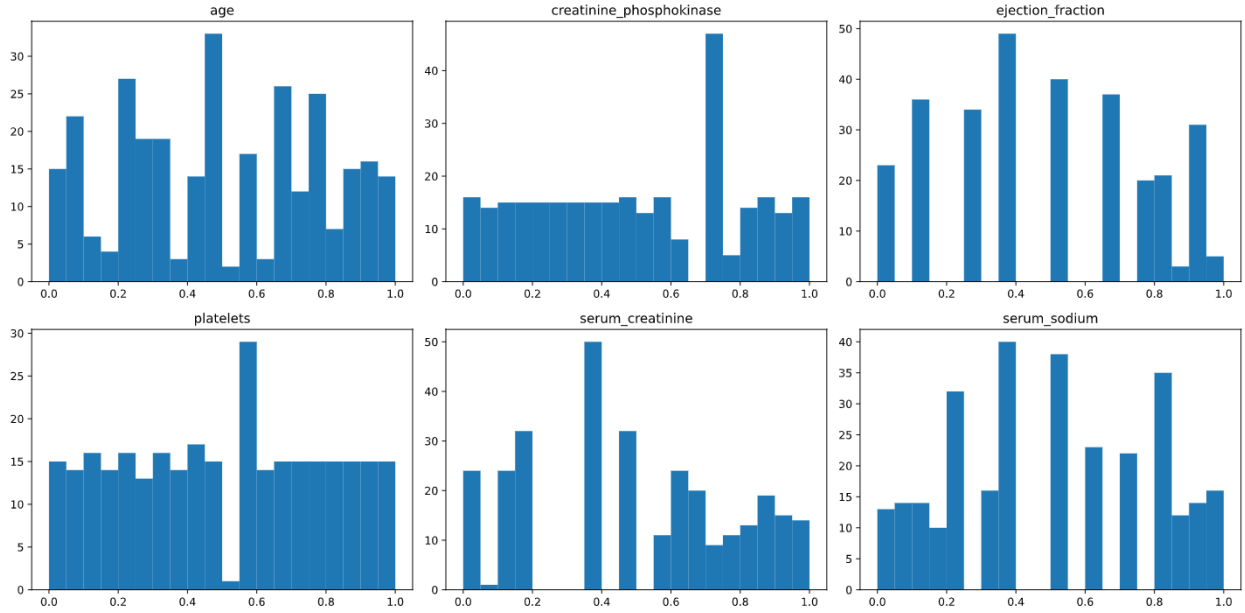

Рисунок 7 – Равномерное распределение

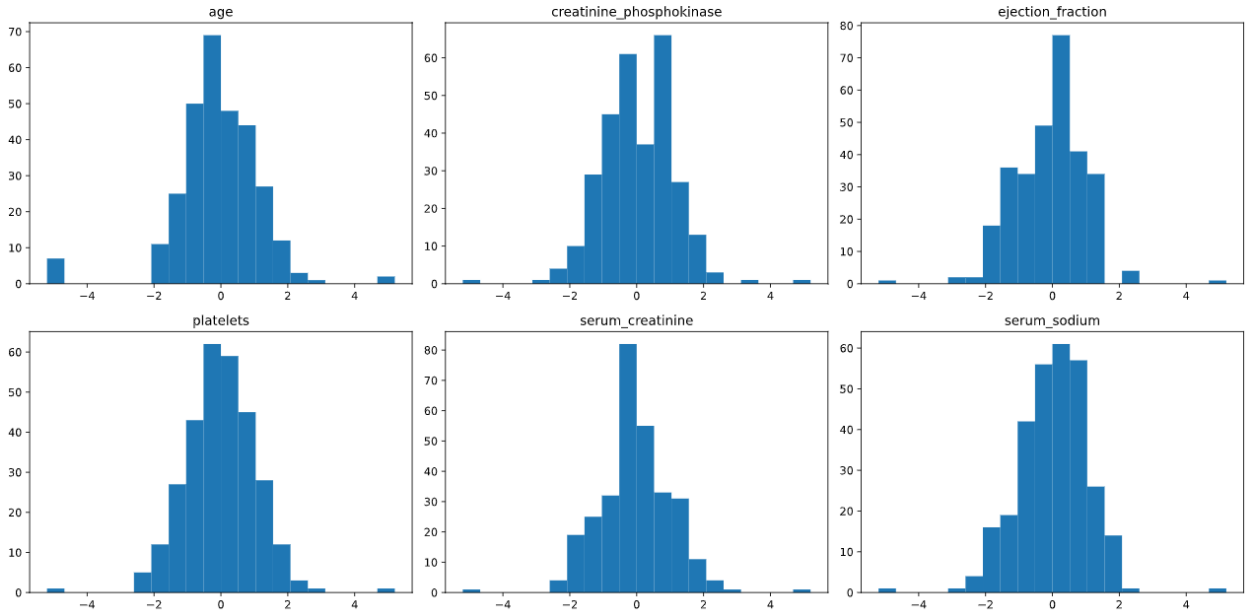

Рисунок 8 – Нормальное распределение (*QuantileTransformer)*

Значение параметра n_*quantiles* в *QuantileTransformer* определяет количество вычисляемых квантилей в ходе настройки. Увеличение повышает частоту дискретизации функции распределения. Количество квантилей не может быть больше, чем количество наблюдений.

2. С помощью *PowerTransformer* данные приведены к нормальному распределению, гистограмма представлены на рис. 9.

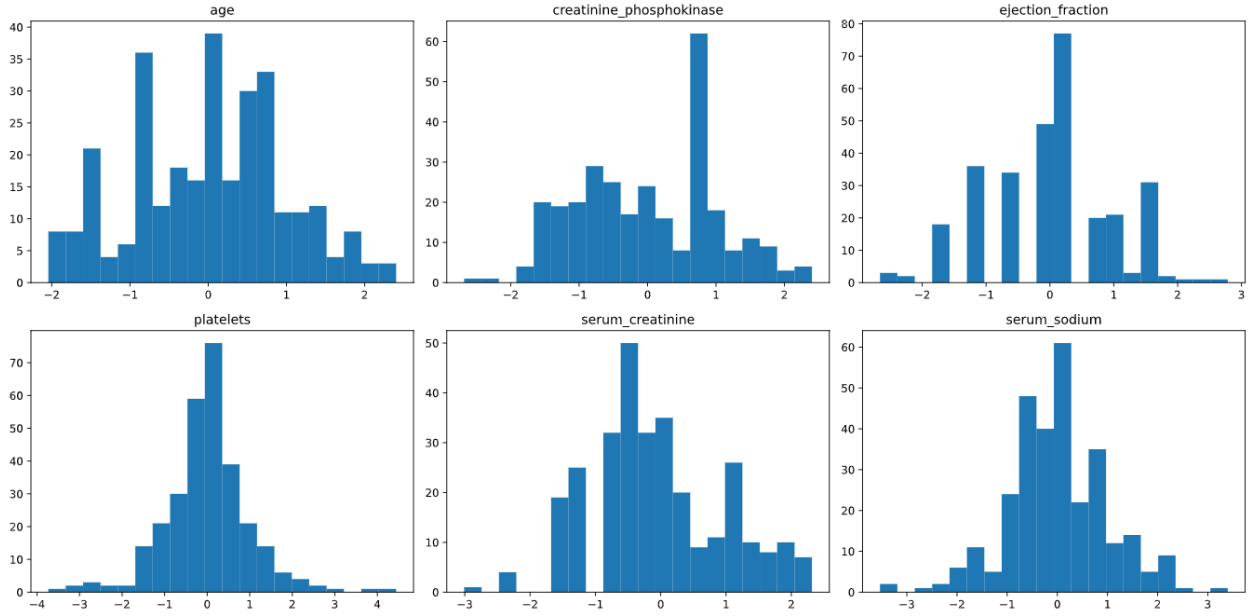

Рисунок 9 – Нормальное распределение (*PowerTransformer*)

Дискретизация признаков

Проведена дискретизация признаков с помощью *KBinsDiscretizer*, результат продемонстрирован на рис. 10. Диапазоны каждого интервала для каждого признака представлены в табл. 5.

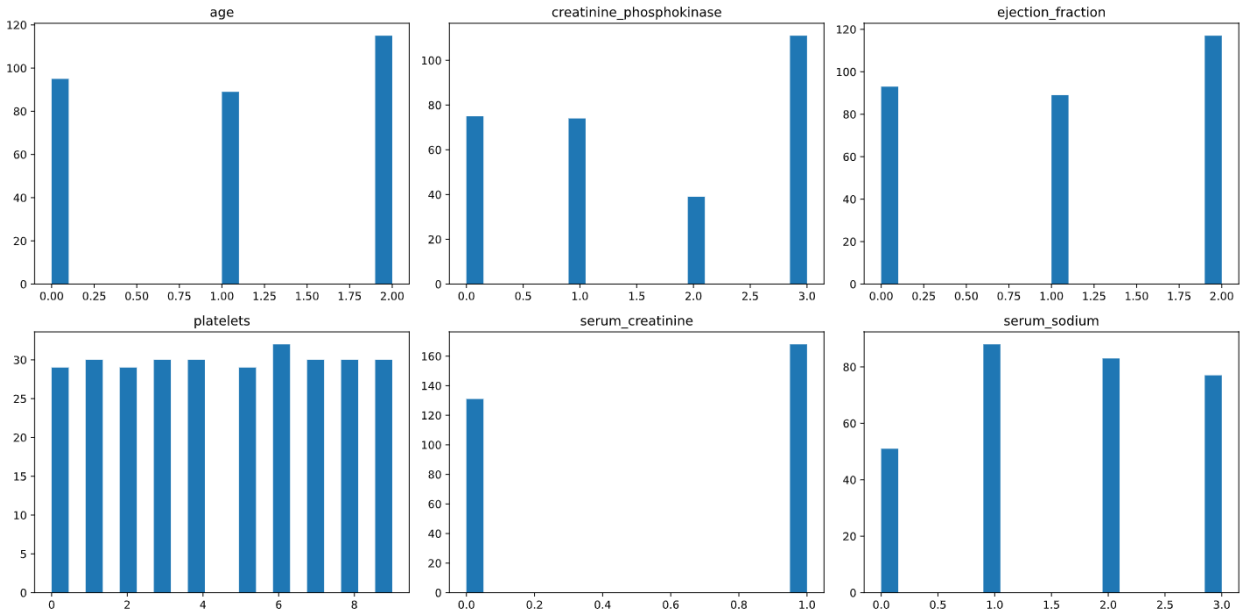

Рисунок 10 – Дискретизация признаков

Таблица 5 – Диапазоны интервалов

age	[40., 55., 65., 95.]
creatinine_phosphokinase	[23. , 116.5, 250. , 582. , 7861.]
ejection_fraction	[14., 35., 40., 80.]
platelets	[25100., 153000., 196000., 221000., 237000., 262000., 265000., 285200., 319800., 374600., 850000.]
serum_creatinine	[0.5, 1.1, 9.4]
serum_sodium	[113., 134., 137., 140., 148.]

Выводы

В ходе лабораторной работы изучены методы предобработки данных из библиотеки *Scikit Learn*:

1. В ходе стандартизации данных установлено, что стандартизация на основе неполного набора наблюдений снижает качество выходных данных.

2. В ходе приведения данных к диапазону построены гистограммы, которые схожи с гистограммами стандартизованных данных. После приведения форма распределения сохраняется.
3. Нелинейные преобразования позволяют привести форму распределения к любой другой форме (равномерная, нормальная и т.д.).
4. Проведена дискретизация данных.

Приложение А

Код программы на python

```
# To add a new cell, type '# %%'
# To add a new markdown cell, type '# %% [markdown]'
null.tpl [markdown]
# # Лабораторная работа №1. Предобработка данных
null.tpl [markdown]
# ## Загрузка данных

# %%
import pandas as pd
import numpy as np
df = pd.read_csv('heart_failure_clinical_records_dataset.csv')
df = df.drop(columns = ['anaemia','diabetes','high_blood_pressure','sex','smoking','t
ime','DEATH_EVENT'])
df

# %%
import matplotlib.pyplot as plt
n_bins = 20
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
# plt.subplots_adjust(left = 0.125, right = 0.9, bottom = 0.1, top = 0.9, wspace=1, h
space=1)
axs[0, 0].hist(df['age'].values, bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(df['creatinine_phosphokinase'].values, bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(df['ejection_fraction'].values, bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(df['platelets'].values, bins = n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(df['serum_creatinine'].values, bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(df['serum_sodium'].values, bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()

# %%
from tabulate import tabulate
min(df['age'].values)
display(tabulate([
        [min(df['age'].values), max(df['age'].values)],
        [min(df['creatinine_phosphokinase'].values), max(df['creatinine_phosphokinase
'].values)],
        [min(df['ejection_fraction'].values), max(df['ejection_fraction'].values)],
        [min(df['platelets'].values), max(df['platelets'].values)],
        [min(df['serum_creatinine'].values), max(df['serum_creatinine'].values)],
        [min(df['serum_sodium'].values), max(df['serum_sodium'].values)]
    ],
    tablefmt='html',
    headers=['min', 'max'],
    showindex=['age', 'creatinine_phosphokinase', 'ejection_fraction', 'platelets', '
serum_creatinine', 'serum_sodium']))

null.tpl [markdown]
# ## Стандартизация данных

# %%
from sklearn import preprocessing
```

```
data = df.values
scaler = preprocessing.StandardScaler().fit(data[:150,:])
data_scaled = scaler.transform(data)
data_scaled_full = preprocessing.StandardScaler().fit_transform(data)

# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
axs[0, 0].hist(data_scaled[:,0], bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_scaled[:,1], bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_scaled[:,2], bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_scaled[:,3], bins = n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_scaled[:,4], bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_scaled[:,5], bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()

# %%
def get_mean_std_var(data):
    return [np.mean(col) for col in data.T], [np.std(col) for col in data.T]

mean_data, std_data = get_mean_std(data)
mean_data_scaled, std_data_scaled = get_mean_std(data_scaled)
mean_data_scaled_full, std_data_scaled_full = get_mean_std(data_scaled_full)

# %%
mean_data, std_data

# %%
mean_data_scaled, std_data_scaled

# %%
mean_data_scaled_full, std_data_scaled_full

# %%
scaler.mean_

# %%
scaler.var_

null.tpl [markdown]
# ## Приведение к диапазону

# %%
min_max_scaler = preprocessing.MinMaxScaler().fit(data)
data_min_max_scaled = min_max_scaler.transform(data)

# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
axs[0, 0].hist(data_min_max_scaled[:,0], bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_min_max_scaled[:,1], bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_min_max_scaled[:,2], bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_min_max_scaled[:,3], bins = n_bins)
axs[1, 0].set_title('platelets')
```

```
axs[1, 1].hist(data_min_max_scaled[:,4], bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_min_max_scaled[:,5], bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()

# %%
min_max_scaler.data_min_

# %%
min_max_scaler.data_max_

# %%
max_abs_scaler = preprocessing.MaxAbsScaler().fit(data)
data_max_abs_scaled = max_abs_scaler.transform(data)

# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
axs[0, 0].hist(data_max_abs_scaled[:,0], bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_max_abs_scaled[:,1], bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_max_abs_scaled[:,2], bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_max_abs_scaled[:,3], bins = n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_max_abs_scaled[:,4], bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_max_abs_scaled[:,5], bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()

# %%
robust_scaler = preprocessing.RobustScaler().fit(data)
data_robust_scaled = robust_scaler.transform(data)

# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
axs[0, 0].hist(data_robust_scaled[:,0], bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_robust_scaled[:,1], bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_robust_scaled[:,2], bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_robust_scaled[:,3], bins = n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_robust_scaled[:,4], bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_robust_scaled[:,5], bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()

# %%
data_min_max_scaled_custom = np.array([[((x - np.min(col)) / (np.max(col) - np.min(co
l))) * 15 - 5 for x in col] for col in data.T]).T
data_min_max_scaled_custom

# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
```

```
axs[0, 0].hist(data_min_max_scaled_custom[:,0], bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_min_max_scaled_custom[:,1], bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_min_max_scaled_custom[:,2], bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_min_max_scaled_custom[:,3], bins = n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_min_max_scaled_custom[:,4], bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_min_max_scaled_custom[:,5], bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()

null.tpl [markdown]
# ## Нелинейные преобразования

# %%
quantile_transformer = preprocessing.QuantileTransformer(n_quantiles = 100, random_st
ate=0).fit(data)
data_quantile_scaled = quantile_transformer.transform(data)

# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
axs[0, 0].hist(data_quantile_scaled[:,0], bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_quantile_scaled[:,1], bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_quantile_scaled[:,2], bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_quantile_scaled[:,3], bins = n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_quantile_scaled[:,4], bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_quantile_scaled[:,5], bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()

# %%
quantile_transformer_normal = preprocessing.QuantileTransformer(n_quantiles = 100, ra
ndom_state=0, output_distribution='normal').fit(data)
data_quantile_scaled_normal = quantile_transformer_normal.transform(data)

# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
axs[0, 0].hist(data_quantile_scaled_normal[:,0], bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_quantile_scaled_normal[:,1], bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_quantile_scaled_normal[:,2], bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_quantile_scaled_normal[:,3], bins = n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_quantile_scaled_normal[:,4], bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_quantile_scaled_normal[:,5], bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()
```

```
# %%
power_transformer = preprocessing.PowerTransformer().fit(data)
data_power_scaled = power_transformer.transform(data)

# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
axs[0, 0].hist(data_power_scaled[:,0], bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_power_scaled[:,1], bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_power_scaled[:,2], bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_power_scaled[:,3], bins = n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_power_scaled[:,4], bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_power_scaled[:,5], bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()

null.tpl [markdown]
# ## Дискретизация признаков

# %%
bins_discretizer = preprocessing.KBinsDiscretizer(n_bins=[3, 4, 3, 10, 2, 4], encode=
'ordinal').fit(data)
data_bins_discretized = bins_discretizer.transform(data)

# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
axs[0, 0].hist(data_bins_discretized[:,0], bins = n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_bins_discretized[:,1], bins = n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_bins_discretized[:,2], bins = n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_bins_discretized[:,3], bins = n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_bins_discretized[:,4], bins = n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_bins_discretized[:,5], bins = n_bins)
axs[1, 2].set_title('serum_sodium')
fig.tight_layout()
plt.show()

# %%
bins_discretizer.bin_edges_

# %%
```