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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Машинное обучение»

Студенты гр. 6304 _____ Тимофеев А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы

Ознакомиться с методами предобработки данных из библиотеки Scikit Learn

Ход работы

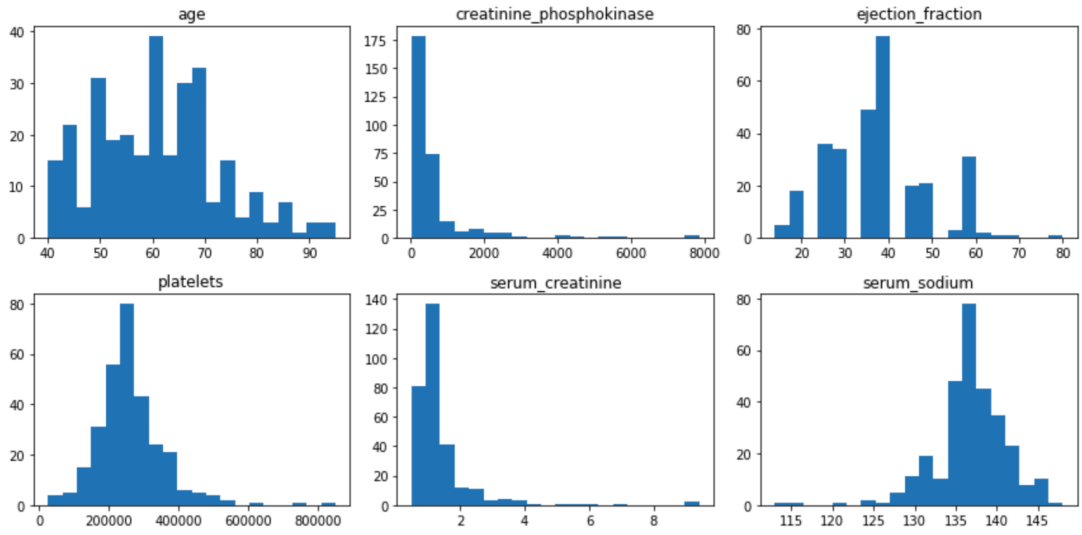
Загрузка данных

1. Был создан датафрейм Pandas на основе загруженного датасета (https://www.kaggle.com/andrewmvd/heart-failure-clinical-data).
2. Из датафрейма были исключены следующие признаки: anaemia, diabetes, high_blood_pressure, sex, smoking, time, DEATH_EVENT.

	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium
count	299.000000	299.000000	299.000000	299.000000	299.00000	299.000000
mean	60.833893	581.839465	38.083612	263358.029264	1.39388	136.625418
std	11.894809	970.287881	11.834841	97804.236869	1.03451	4.412477
min	40.000000	23.000000	14.000000	25100.000000	0.50000	113.000000
25%	51.000000	116.500000	30.000000	212500.000000	0.90000	134.000000
50%	60.000000	250.000000	38.000000	262000.000000	1.10000	137.000000
75%	70.000000	582.000000	45.000000	303500.000000	1.40000	140.000000
max	95.000000	7861.000000	80.000000	850000.000000	9.40000	148.000000

Рисунок 1 – Описание полученного датафрейма

3. Были построены гистограммы признаков, определены диапазоны значений для каждого признака, а также максимальные значения и промежутки, в которых они наблюдались.

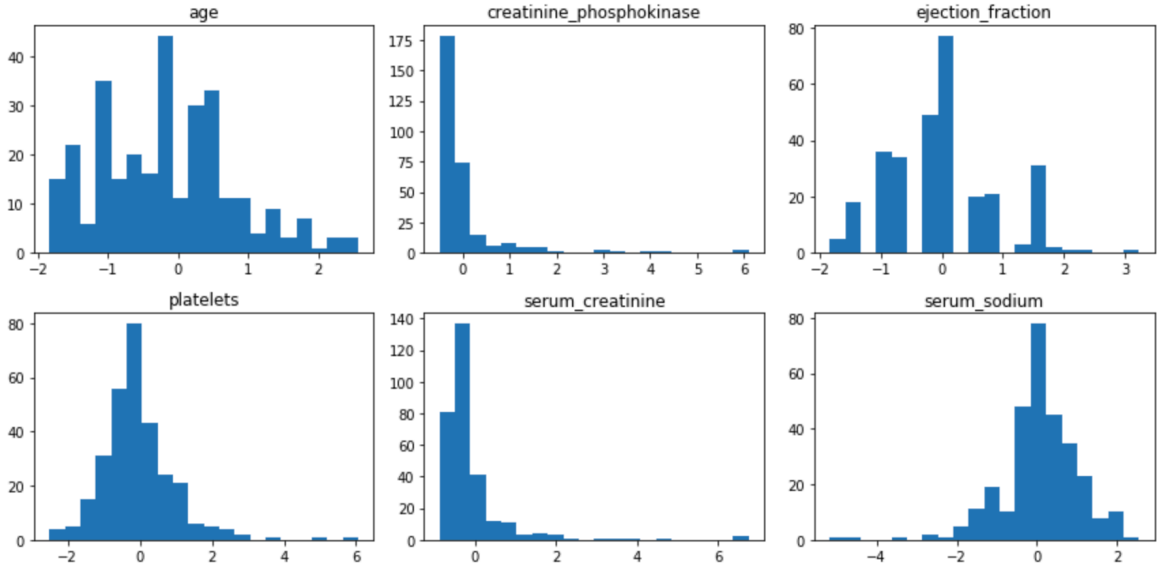



```
1. min: 40.00, max: 95.00, макс. наблюдений на промежутке: 59.25 – 62.00(наблюдений: 39.0)
2. min: 23.00, max: 7861.00, макс. наблюдений на промежутке: 23.00 – 414.90(наблюдений: 178.0)
3. min: 14.00, max: 80.00, макс. наблюдений на промежутке: 37.10 – 40.40(наблюдений: 77.0)
4. min: 25100.00, max: 850000.00, макс. наблюдений на промежутке: 231325.00 – 272570.00(наблюдений: 80.0)
5. min: 0.50, max: 9.40, макс. наблюдений на промежутке: 0.95 – 1.39(наблюдений: 137.0)
6. min: 113.00, max: 148.00, макс. наблюдений на промежутке: 135.75 – 137.50(наблюдений: 78.0)
```

Рисунок 2 – Гистограммы признаков

Стандартизация данных

1. Была произведена стандартизация на 150 значениях, построены гистограммы полученных данных.




```
1. min: -1.84, max: 2.57, макс. наблюдений на промежутке: -0.30 – -0.08(наблюдений: 44.0)
2. min: -0.49, max: 6.10, макс. наблюдений на промежутке: -0.49 – -0.16(наблюдений: 178.0)
3. min: -1.84, max: 3.23, макс. наблюдений на промежутке: -0.06 – 0.19(наблюдений: 77.0)
4. min: -2.51, max: 6.06, макс. наблюдений на промежутке: -0.37 – 0.06(наблюдений: 80.0)
5. min: -0.87, max: 6.76, макс. наблюдений на промежутке: -0.49 – -0.11(наблюдений: 137.0)
6. min: -5.17, max: 2.54, макс. наблюдений на промежутке: -0.15 – 0.23(наблюдений: 78.0)
```

Рисунок 3 – Гистограмма данных, стандартизированных на 150 значениях

2. Было выполнено сравнение математического ожидания и СКО для исходных данных, данных, стандартизированных на 150 значений, а также данных, стандартизированных на всей выборке. В сравнении были рассмотрены значения из атрибутов скейлера mean_ и scale_. Результаты сравнения представлены в таблице 1.

Таблица 1 – Сравнение мат. Ожидания и СКО

	age	creatinine_phospho kinase	ejection_fraction	platelets	serum_creatinine	serum_sodium
Среднее	*60.83*	*581.84*	*38.08*	*263358.03*	*1.39*	*136.63*
СКО	*11.87*	*968.66*	*11.82*	*97640.55*	*1.03*	*4.41*
Среднее (ст. 150)	*-0.17*	*-0.02*	*0.01*	*-0.04*	*-0.11*	*0.04*
СКО (ст. 150)	*0.95*	*0.81*	*0.91*	*1.02*	*0.89*	*0.97*

Среднее (ст. 150 mean_)	*62.95*	*607.15*	*37.95*	*266746.75*	*1.52*	*136.45*
СКО (ст. 150 scale_)	*12.45*	*1189.74*	*13.04*	*96191.79*	*1.17*	*4.54*
Среднее (ст. все)	*0.00*	*0.00*	*-0.00*	*0.00*	*0.00*	*-0.00*
СКО (ст. все)	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*	*1.00*
Среднее (ст. все mean_)	*60.83*	*581.84*	*38.08*	*263358.03*	*1.39*	*136.63*
СКО (ст. все scale_)	*11.87*	*968.66*	*11.82*	*97640.55*	*1.03*	*4.41*

На основании полученных данных можно сделать вывод, что StandardScaler приводит исходную выборку к выборке с нулевым мат. ожиданием и единичной дисперсией, по формуле:

$$X_i^{`} = \frac{X_i - M[X]}{\sqrt{D[X]}}$$

где X_i – значения исходной выборки, а $X_i^{`}$ – результат. В атрибутах mean_ и scale_ скейлера хранятся значения мат. ожидания и СКО исходной выборки, которые используются при стандартизации.

Приведение к диапазону

1. Было выполнено приведение к диапазону при помощи MinMaxScaler, построены гистограммы признаков.

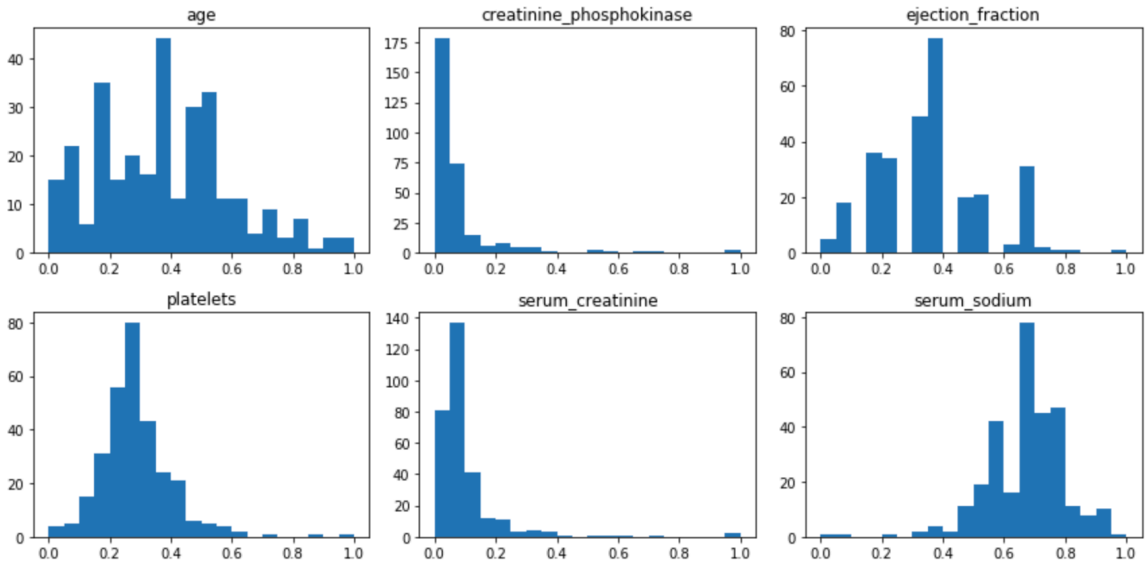


Рисунок 4 – Гистограммы после использования MinMaxScaler

Данные были приведены к диапазону [0, 1] по формуле:

$$X_i^{`} = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$

где X_i – значения исходной выборки, а $X_i^{`}$ – результат.

2. Из атрибутов data_min_ и data_max_ были получены минимальные и максимальные значения признаков. Результаты представлены в таблице 2.

Таблица 2 – Минимальные и максимальные значения признаков.

	age	creatinine_phospho kinase	ejection_fraction	platelets	serum_creatinine	serum_sodium
Min	*40*	*23*	*14*	*25100*	*0.50*	*113*
Max	*95*	*7861*	*80*	*850000*	*9.40*	*148*

3. Были выполнены трансформации с помощью MaxAbsScaler и RobustScaler.

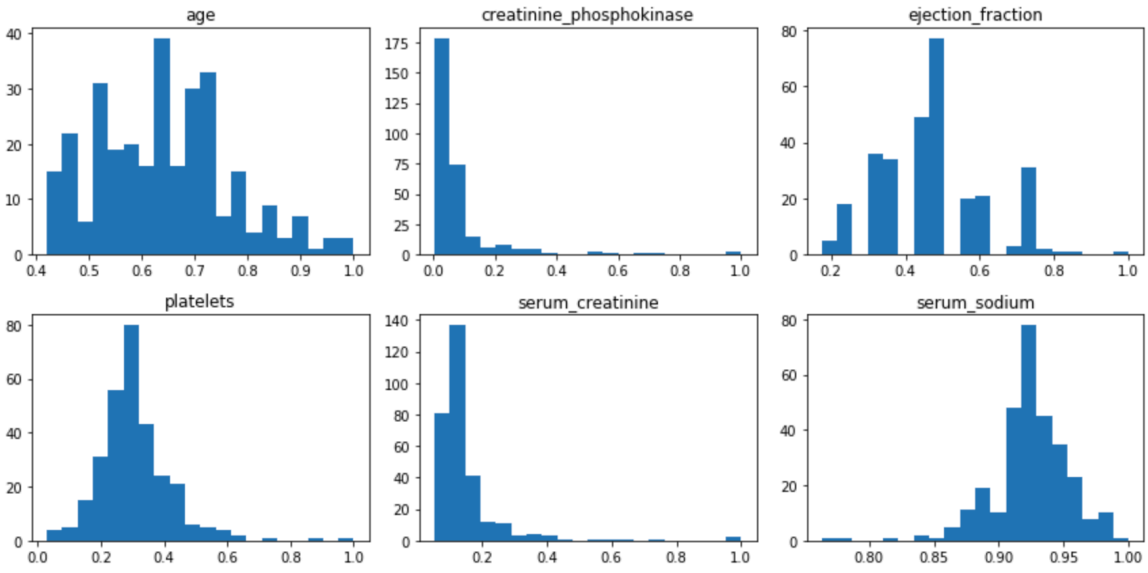


Рисунок 5 – Гистограммы после использования MaxAbsScaler

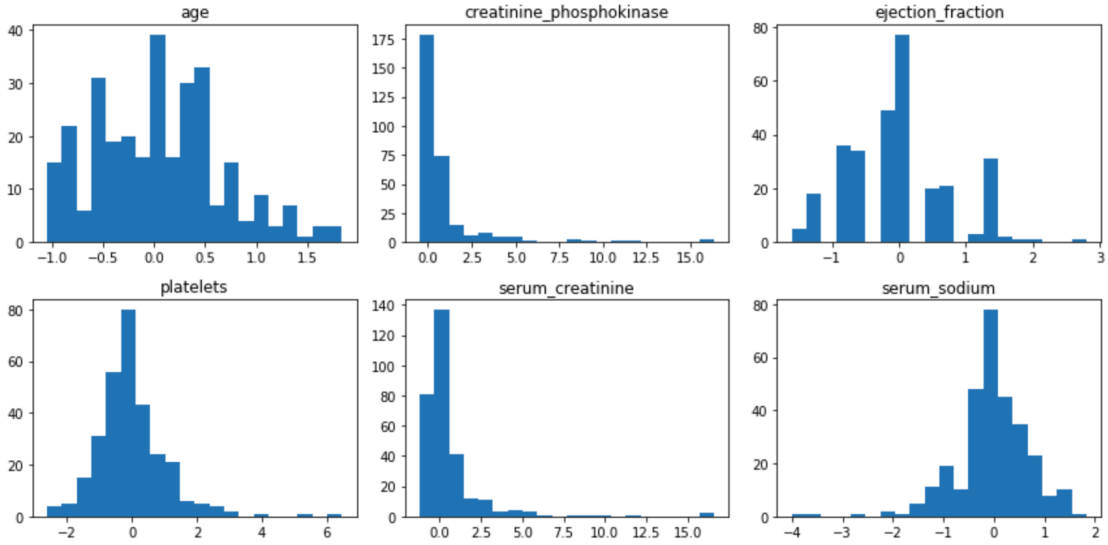


Рисунок 6 – Гистограммы после использования RobustScaler

MaxAbsScaler приводит к диапазону таки образом, чтобы максимальное абсолютное значение выборки было равно единице. RobustScaler приводит медиану выборки к 0, а также масштабирует выборку относительно межквартильного диапазона.

4. Была написана функция, приводящая выборку к диапазону [-5, 10]:

```
def fit_5_10(data):
    scaler = preprocessing.MinMaxScaler(feature_range=(-5, 10)).fit(data)
    return scaler.transform(data)
```

Нелинейные преобразования

1. Было выполнено приведение данных к равномерному и нормальному распределениям при помощи QuantileTransformer.

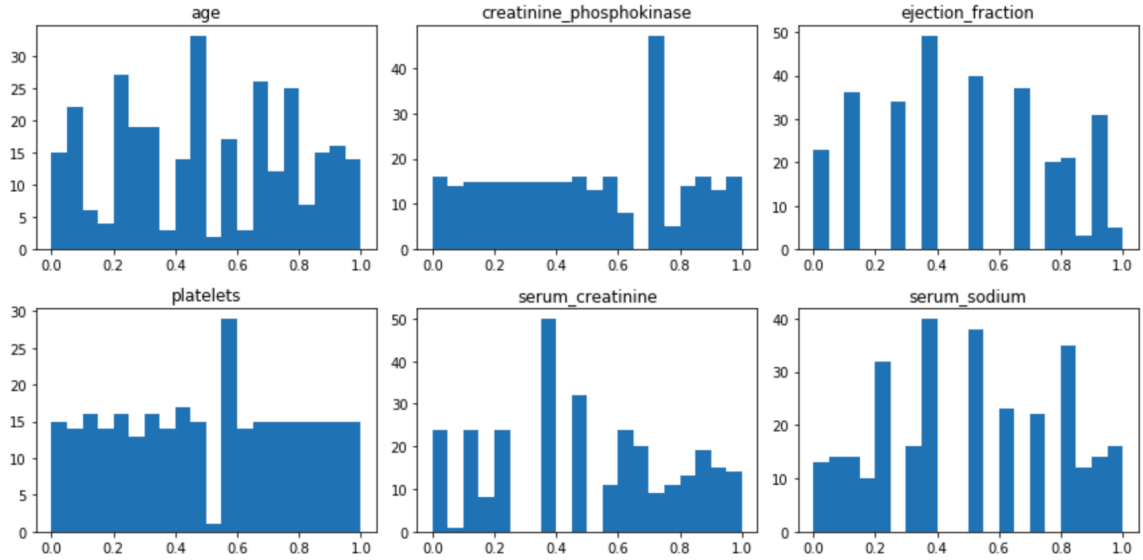


Рисунок 7 – Гистограммы после использования QuantileTransformer (равн.)

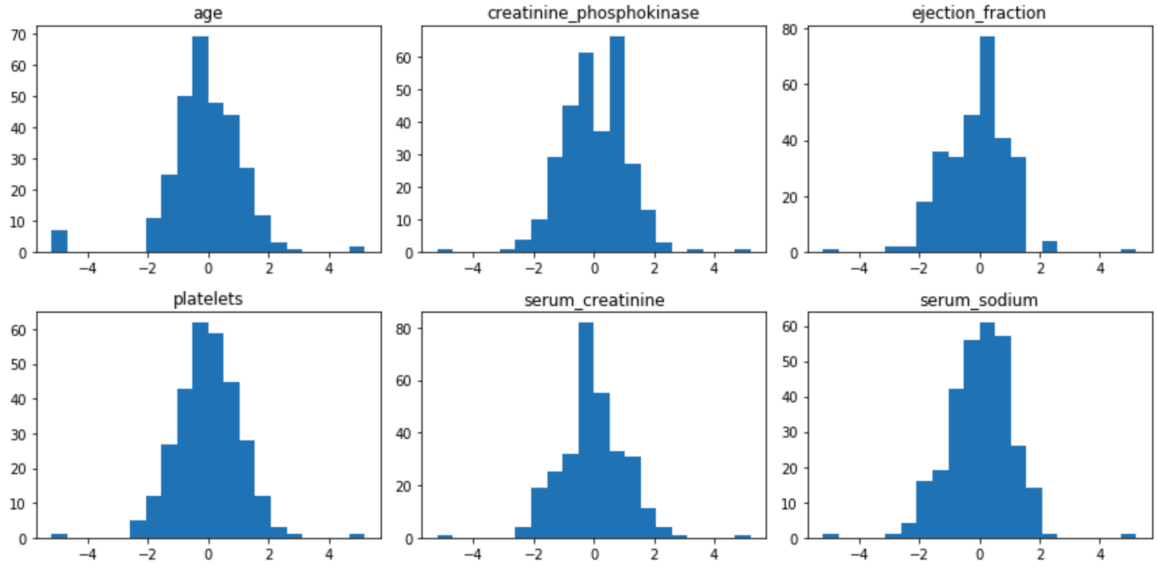


Рисунок 8 – Гистограммы после использования QuantileTransformer (норм.)

Задаваемое количество квантилей определяет их число при дискретизации функции распределения. Чем больше значение, тем распределение результирующей выборки ближе к требуемому.

2. Было выполнено приведение данных к нормальному распределению при помощи PowerTransformer.

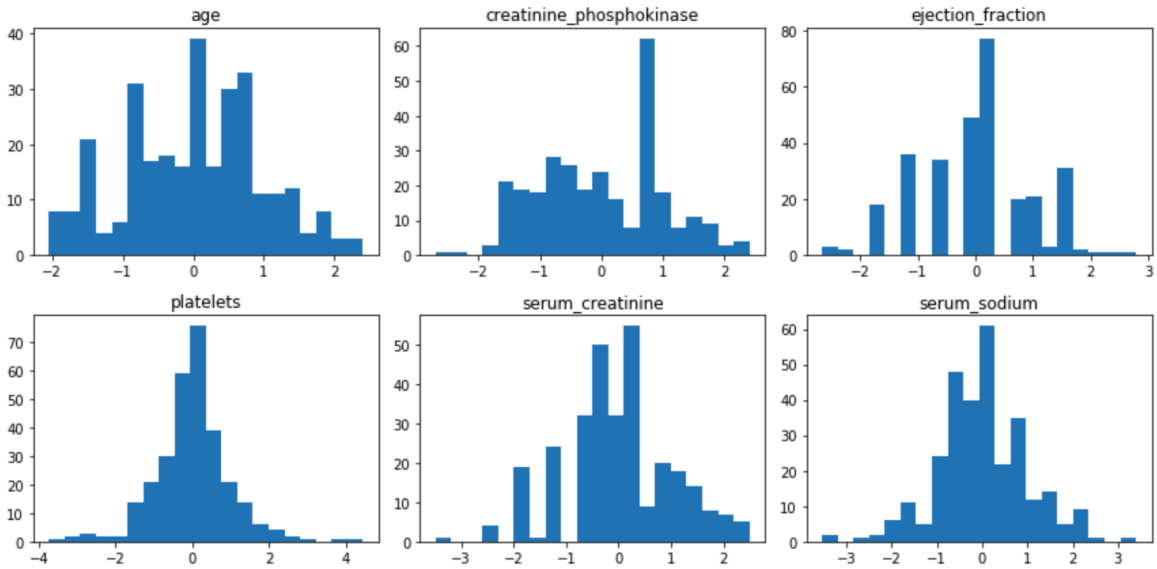


Рисунок 9 – Гистограммы после использования PowerTransformer

Дискретизация признаков

1. Была проведена дискретизация признаков при помощи KBinsDiscretizer.

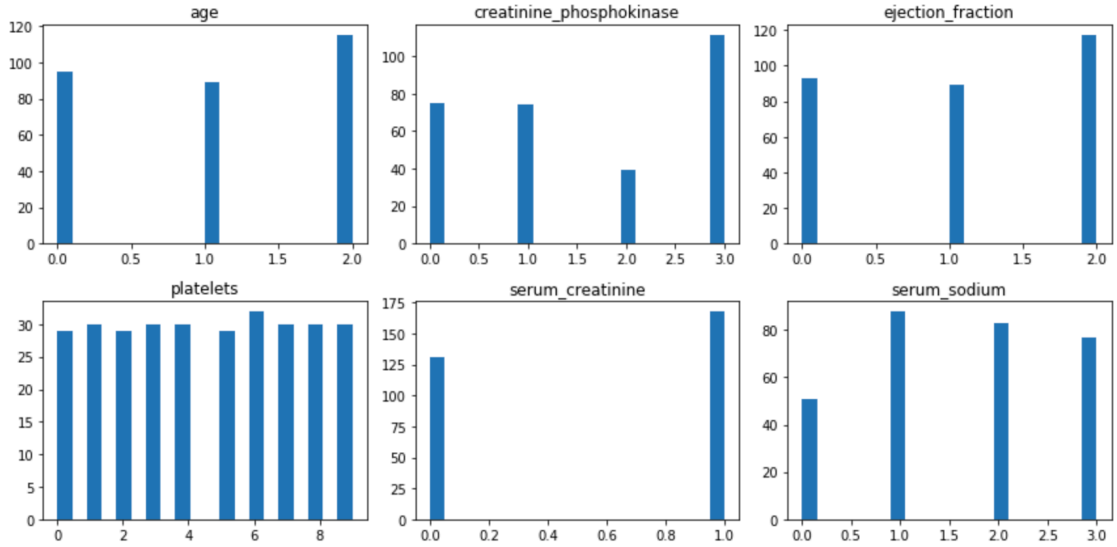


Рисунок 9 – Гистограммы после дискретизации

Через атрибут bin_edges_ были получены диапазоны интервалов для каждого признака. Результаты представлены в таблице 3.

Таблица 3 – Диапазоны интервалов признаков

Признак	Диапазоны
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

Было выполнено знакомство с методами предобработки данных из библиотеки Scikit Learn.

Полученные результаты показали:

- Количество значений в выборке влияет на качество стандартизации
- Приведение к диапазону не изменяет форму выборки
- Нелинейные преобразования изменяют форму выборки в соответствии с выбранным распределением

Использованный код представлен в приложении А.

ПРИЛОЖЕНИЕ А

Исходный код

```
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','time','DEATH_EVENT'])

#Вывод датафрейма с данными для лаб. работы. Должно быть 299 наблюдений и 6 признаков
df.describe()

import matplotlib.pyplot as plt

def minMaxBinAndPlot(data, bins=20):
    fig, axs = plt.subplots(2,3, figsize=(12, 6))
    hists = []
    hists.append(axs[0, 0].hist(data[:,0], bins = bins))
    axs[0, 0].set_title('age')

    hists.append(axs[0, 1].hist(data[:,1], bins = bins))
    axs[0, 1].set_title('creatinine_phosphokinase')

    hists.append(axs[0, 2].hist(data[:,2], bins = bins))
    axs[0, 2].set_title('ejection_fraction')

    hists.append(axs[1, 0].hist(data[:,3], bins = bins))
    axs[1, 0].set_title('platelets')

    hists.append(axs[1, 1].hist(data[:,4], bins = bins))
    axs[1, 1].set_title('serum_creatinine')

    hists.append(axs[1, 2].hist(data[:,5], bins = bins))
    axs[1, 2].set_title('serum_sodium')

    plt.tight_layout()
    plt.show()

    for index in range(len(hists)):
        mostObservations = np.argmax(hists[index][0])
        print('{}. min: {:.2f}, max: {:.2f}, макс. наблюдений на промежутке: {:.2f} -
{:.2f}(наблюдений: {})'.format(index + 1, hists[index][1][0], hists[index][1][bins],
hists[index][1][mostObservations], hists[index][1][mostObservations + 1],
hists[index][0][mostObservations]))

data = df.to_numpy(dtype='float')
minMaxBinAndPlot(data)

from sklearn import preprocessing

scaler = preprocessing.StandardScaler().fit(data[:150,:])
data_scaled = scaler.transform(data)

minMaxBinAndPlot(data_scaled)

def meanAndStd(data):
    for col in data.T:
        print('mean: {:.2f}, std: {:.2f}'.format(np.mean(col), np.std(col)))
# стандартизация на всех данных
all_scaler = preprocessing.StandardScaler().fit(data)
```

```
all_data_scaled = all_scaler.transform(data)
print('Исходные данные')
meanAndStd(data)
print('Стандартизация на 150')
meanAndStd(data_scaled)
print('Стандартизация на всех')
meanAndStd(all_data_scaled)

print('Значение из mean_ и scale_:')
print('Стандартизация на 150')
for i in range(6):
    print('mean: {:.2f}, std: {:.2f}'.format(scaler.mean_[i], scaler.scale_[i]))
print('Стандартизация на всех')
for i in range(6):
    print('mean: {:.2f}, std: {:.2f}'.format(all_scaler.mean_[i],
all_scaler.scale_[i]))

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

minMaxBinAndPlot(data_min_max_scaled)

for i in range(6):
    print('min: {:.2f}, max: {:.2f}'.format(min_max_scaler.data_min_[i],
min_max_scaler.data_max_[i]))

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

minMaxBinAndPlot(data_max_abs_scaled)

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

minMaxBinAndPlot(data_robust_scaled)

def fit_5_10(data):
    scaler = preprocessing.MinMaxScaler(feature_range=(-5, 10)).fit(data)
    return scaler.transform(data)

quantile_transformer = preprocessing.QuantileTransformer(n_quantiles = 100,
random_state=0, output_distribution='normal').fit(data)
data_quantile_scaled = quantile_transformer.transform(data)

minMaxBinAndPlot(data_quantile_scaled)

power_transformer = preprocessing.PowerTransformer(method='box-cox').fit(data)
data_power_transformed = power_transformer.transform(data)

minMaxBinAndPlot(data_power_transformed)

discretizer = preprocessing.KBinsDiscretizer(n_bins=[3, 4, 3, 10, 2, 4],
encode='ordinal').fit(data)
data_disc = discretizer.transform(data)

print(discretizer.bin_edges_)

minMaxBinAndPlot(data_disc)
```