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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Машинное обучение»

Тема: Предобработка данных

Студент гр. 1310 _____ Комаров Д. Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2025

Постановка задачи

Цель работы: Ознакомление с методами предобработки данных из библиотеки Scikit Learn.

Выполнение лабораторной работы

Загрузка данных

Для выполнения лабораторной работы используем набор данных «Heart Failure Prediction», из которого возьмем только 6 признаков: *age*, *creatinine_phosphokinase*, *ejection_fraction*, *platelets*, *serum_creatinine* и *serum_sodium*. Для каждого из признаков в данном наборе данных содержится 299 наблюдений. На рисунке 1 представлены гистограммы распределений признаков.

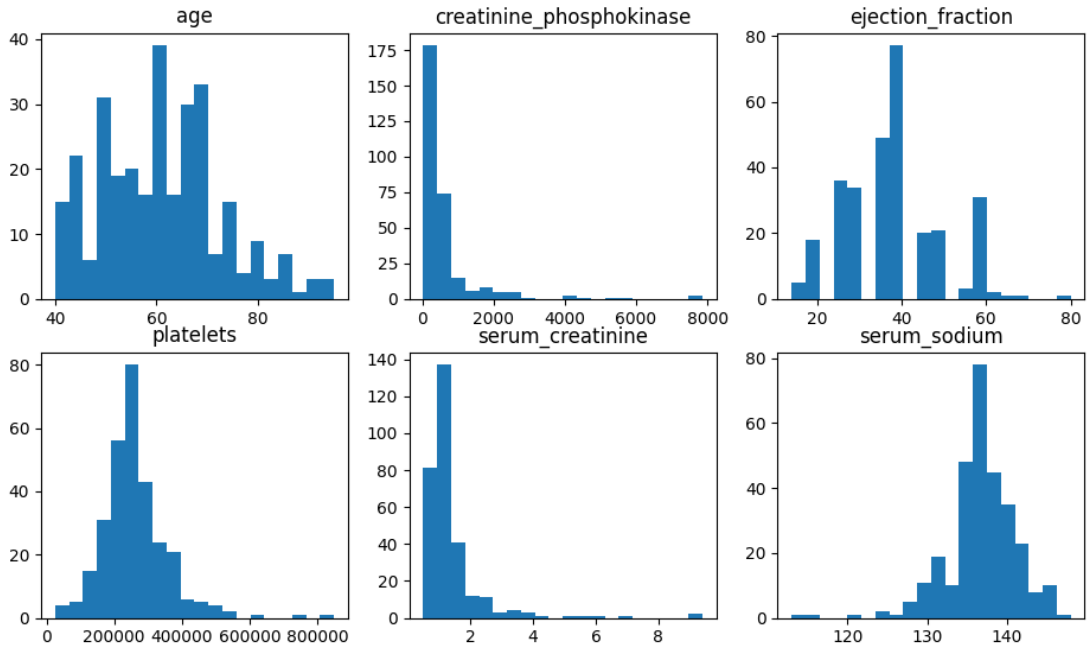

Рисунок 1 – Гистограммы распределений признаков

В таблице 1 для каждого из признаков приведены диапазон, математическое ожидание, среднеквадратичное отклонение.

Таблица 1 – Параметры распределений признаков

Признак	*age*	*creatinine phosphokinase*	*ejection fraction*	*platelets*	*creatinine*	*serum sodium*
Диапазон	[40;95]	[23;7861]	[14;80]	[25100; 850000]	[0.5;9.4]	[113;148]

Продолжение таблицы 1

Математическое ожидание	60.83	581.84	38.08	263358.03	1.39	136.63
Среднеквадратичное отклонение	11.87	968.66	11.82	97640.55	1.03	4.41

Стандартизация данных

Стандартизируем данные на основе первых 150 наблюдений при помощи библиотеки Sklearn. На рисунке 2 представлены гистограммы распределений признаков после стандартизации.

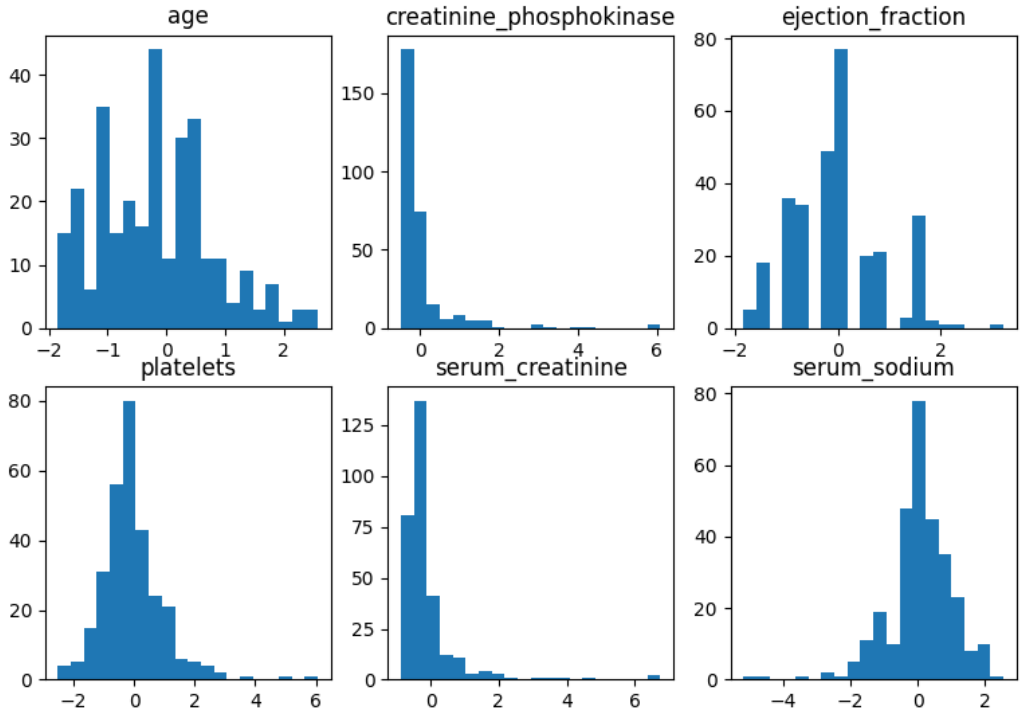

Рисунок 2 – Гистограммы распределений признаков после стандартизации

В таблице 2 для каждого из признаков приведены диапазон, математическое ожидание, среднеквадратичное отклонение после стандартизации.

Таблица 2 – Параметры распределений признаков после стандартизации

Признак	age	creatinine phosphokinase	ejection fraction	platelets	creatinine	serum sodium
Диапазон	[-1.84; 2.57]	[-0.49;6.1]	[-1.84; 3.23]	[-2.51; 6.06]	[-0.87; 6.76]	[-5.17; 2.54]

Продолжение таблицы 2

Математическое ожидание	-0.17	-0.02	0.01	-0.04	-0.11	0.04
Среднеквадратичное отклонение	0.95	0.81	0.91	1.02	0.89	0.97

Отметим, что в ходе проведения стандартизации согласно таблицам 1 и 2 математические ожидания признаков стали близки к 0, а среднеквадратичные отклонения к 1. Также заметим, что диапазоны значений для всех признаков после стандартизации стали сопоставимыми друг с другом.

При помощи таблиц 1 и 2 выведем формулы, по которым проводилась стандартизация значений. Пусть *i* – номер наблюдения. Тогда для признака *age* формула стандартизации была

$$x_{i\,st} = \frac{x_i - 61}{12.49},$$

для признака *creatinine_phosphokinase*

$$x_{i\,st} = \frac{x_i - 581.86}{1195.88},$$

для признака *ejection_fraction*

$$x_{i\,st} = \frac{x_i - 38.07}{12.98},$$

для признака *platelets*

$$x_{i\,st} = \frac{x_i - 263358.07}{95726.03},$$

для признака *creatinine*

$$x_{i\,st} = \frac{x_i - 1.5}{1.15},$$

для признака *serum_sodium*

$$x_{i\,st} = \frac{x_i - 136.59}{4.55}.$$

Сравним полученные формулы со значениями полей *mean_* и *var_* объекта *scaler*, проводившего стандартизацию. Значения этих полей представлены в таблице 3.

Таблица 3 – Значения полей объекта *scaler*

Признак	*age*	*creatinine phosphokinase*	*ejection fraction*	*platelets*	*creatinine*	*serum sodium*
*mean_*	62.95	607.15	37.95	266746.75	1.52	136.45
*var_*	12.45	1189.74	13.04	96191.79	1.17	4.54

Параметры вычисленных вручную формул оказались аналогичны параметрам объекта *scaler*.

Проведем стандартизацию на всех данных. На рисунке 3 представлены гистограммы распределений признаков после стандартизации на всех данных.

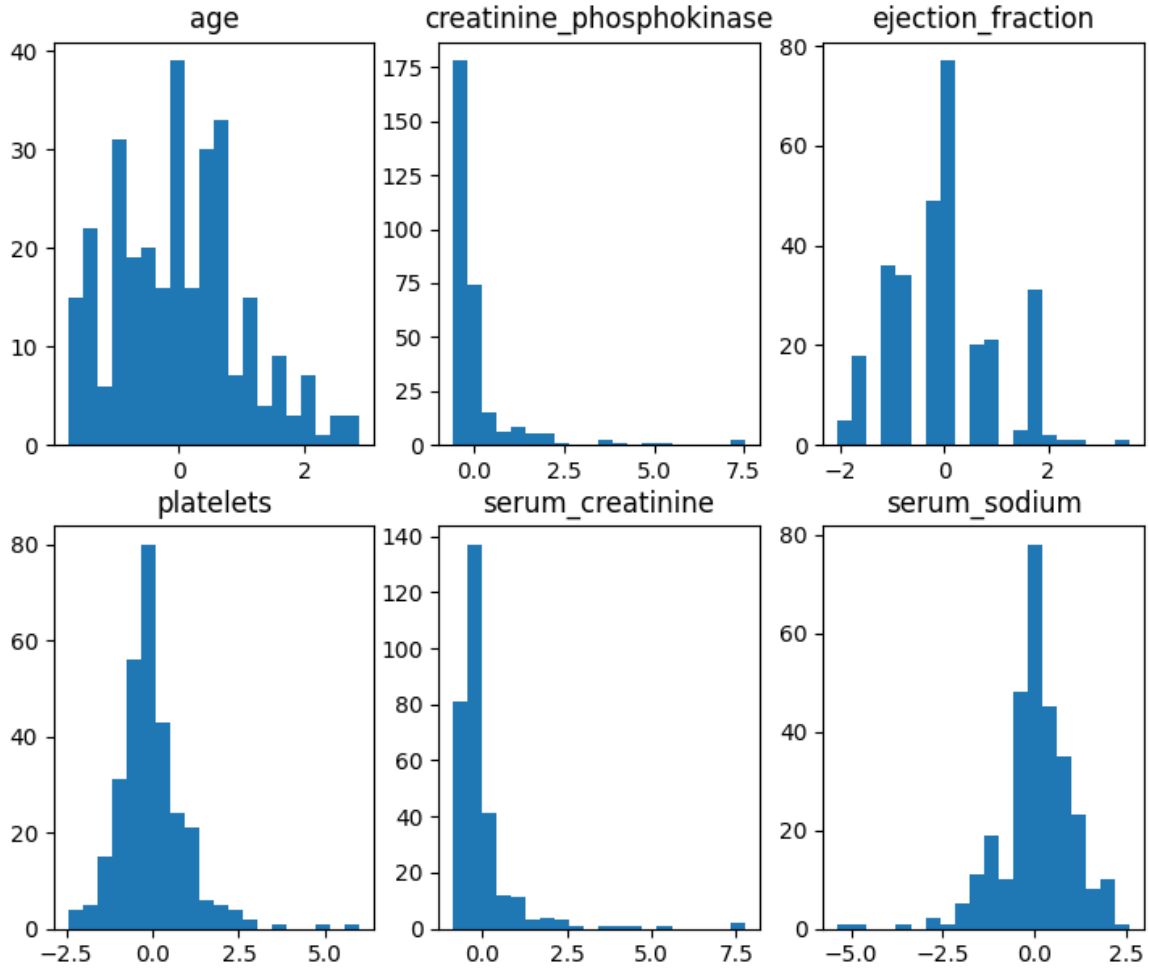

Рисунок 3 – Гистограммы распределений признаков после стандартизации на всех данных

В таблице 4 для каждого из признаков приведены диапазон, математическое ожидание, среднеквадратичное отклонение после стандартизации на всех данных.

Таблица 4 – Параметры распределений признаков после стандартизации на всех данных

Признак	*age*	*creatinine phospho kinase*	*ejection fraction*	*platelets*	*creatinine*	*serum sodium*
Диапазон	[-1.75; 2.88]	[-0.58; 7.51]	[-2.04; 3.55]	[-2.44; 6.01]	[-0.87; 7.75]	[-5.36; 2.58]
Математическое ожидание	0	0	0	0	0	0
Среднеквадратичное отклонение	1	1	1	1	1	1

Из рисунка 3 и таблицы 4 видно, что при стандартизации по всем данным гистограммы и диапазоны значений изменились незначительно по сравнению со стандартизацией на основе первых 150 наблюдений. Математические ожидания и среднеквадратичные отклонения же стали равны 0 и 1 соответственно для всех признаков.

Приведение к диапазону

Приведем данные к диапазону, используя *MinMaxScaler.* На рисунке 4 представлены гистограммы распределений признаков после приведения к диапазону.

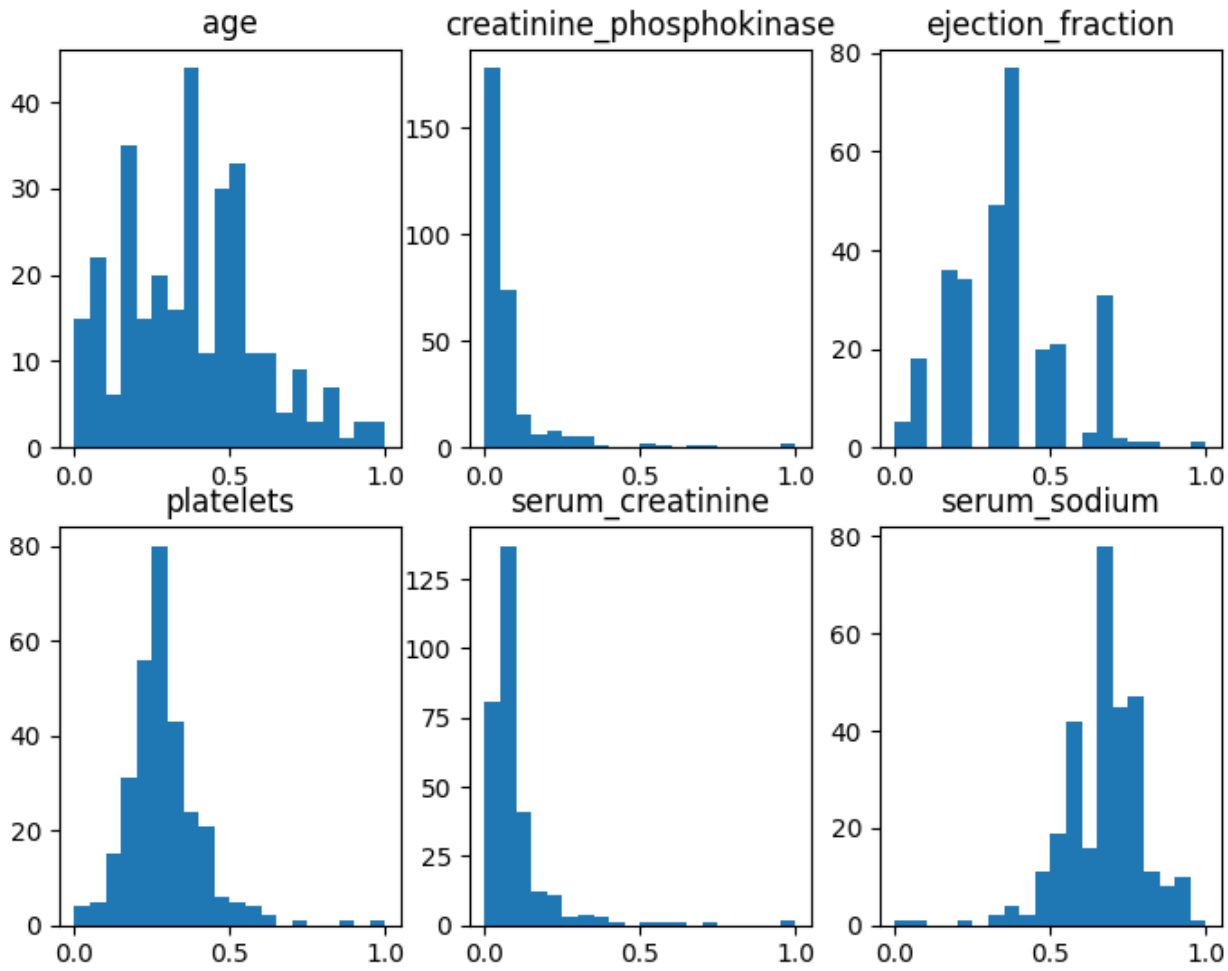

Рисунок 4 – Гистограммы распределений признаков после приведения данных к диапазону

Из гистограмм, представленных на рисунке 4 видно, что, после приведения к диапазону, все распределения стали иметь диапазон [0;1].

Параметры объекта *MinMaxScaler* представлены в таблице 5.

Таблица 5 – Параметры объекта *MinMaxScaler*

Признак	*age*	*creatinine phosphokinase*	*ejection fraction*	*platelets*	*creatinine*	*serum sodium*
*data_min_*	40	23	14	25100	0.5	113
data_max_:	95	7861	80	850000	9.4	148

Из таблицы 5 видно, что минимальное и максимальное значение каждого признака в объекте *MinMaxScaler* совпадает с таковым, вычисленным ранее.

На рисунке 5 представлены гистограммы распределений признаков после приведения к диапазону при помощи *MaxAbsScaler*.

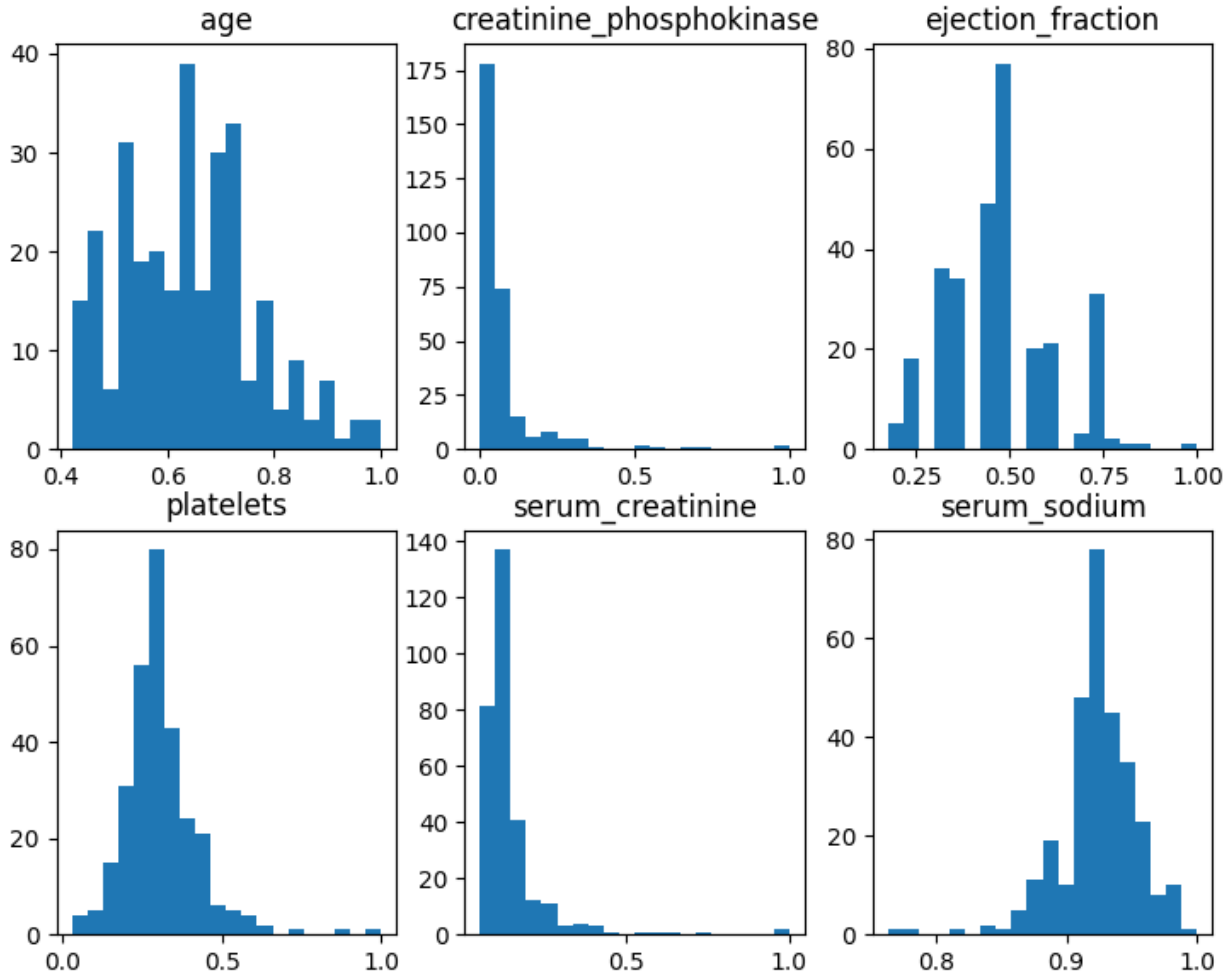

Рисунок 5 – Гистограммы распределений признаков после приведения данных к диапазону при помощи *MaxAbsScaler*

После приведения диапазону при помощи *MaxAbsScaler* данные каждого признака были поделены на число, равное максимуму из модулей данных каждого признака, после чего правая граница диапазонов всех признаков стала равна 1.

На рисунке 6 представлены гистограммы распределений признаков после приведения к диапазону при помощи *RobustScaler*.

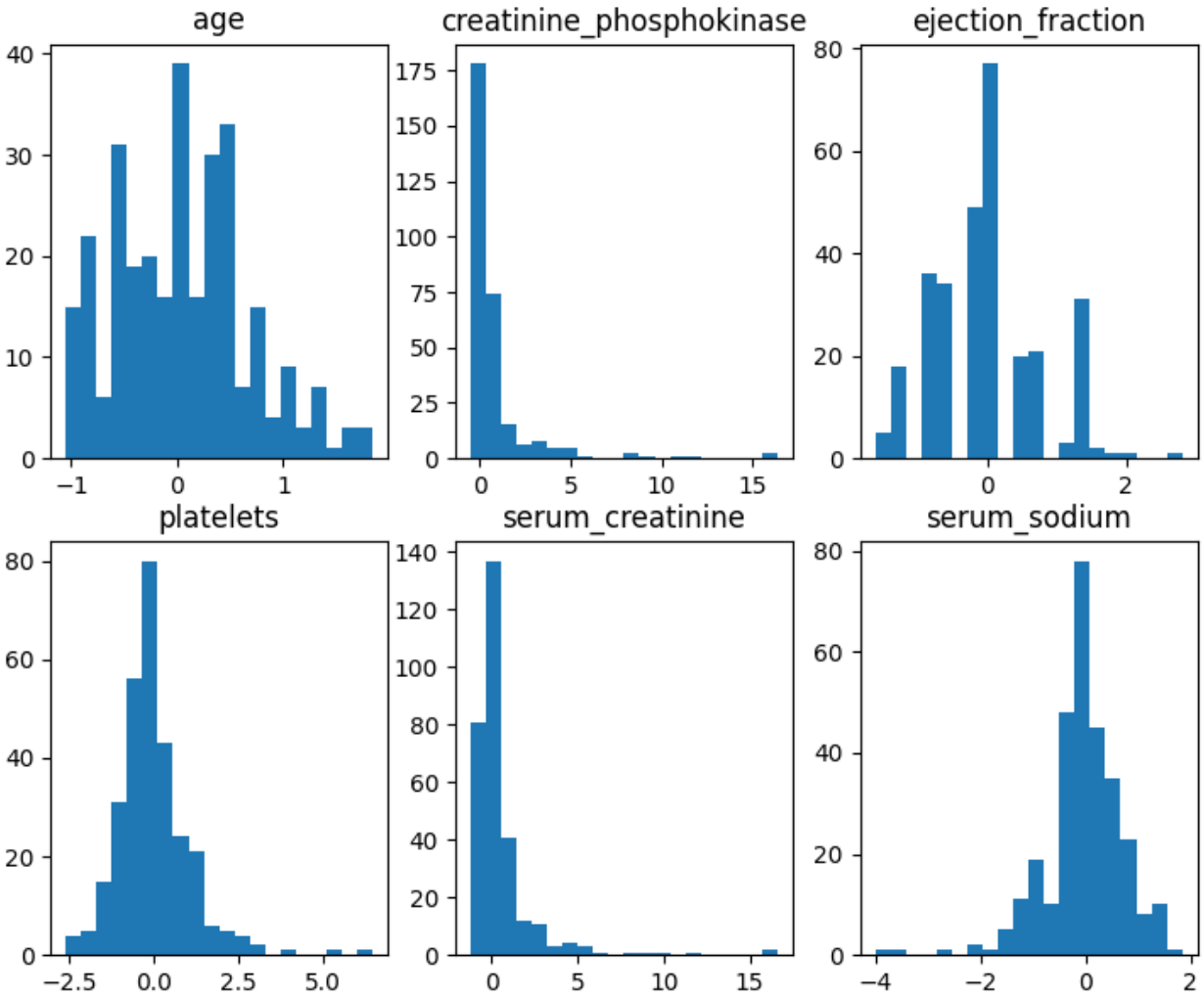

Рисунок 6 – Гистограммы распределений признаков после приведения данных к диапазону при помощи *RobustScaler*

После приведения диапазону при помощи *RobustScaler* из данных каждого признака была вычтена медиана данного признака, после чего было произведено деление на величину межквартильного размаха.

Напишем функцию, которая приводит все данные к диапазону [-5;10]. Код данной функции вместе со всем остальным кодом данной лабораторной работы представлен в приложении А. На рисунке 7 представлены гистограммы распределений признаков после приведения к диапазону [-5;10] при помощи написанной функции.

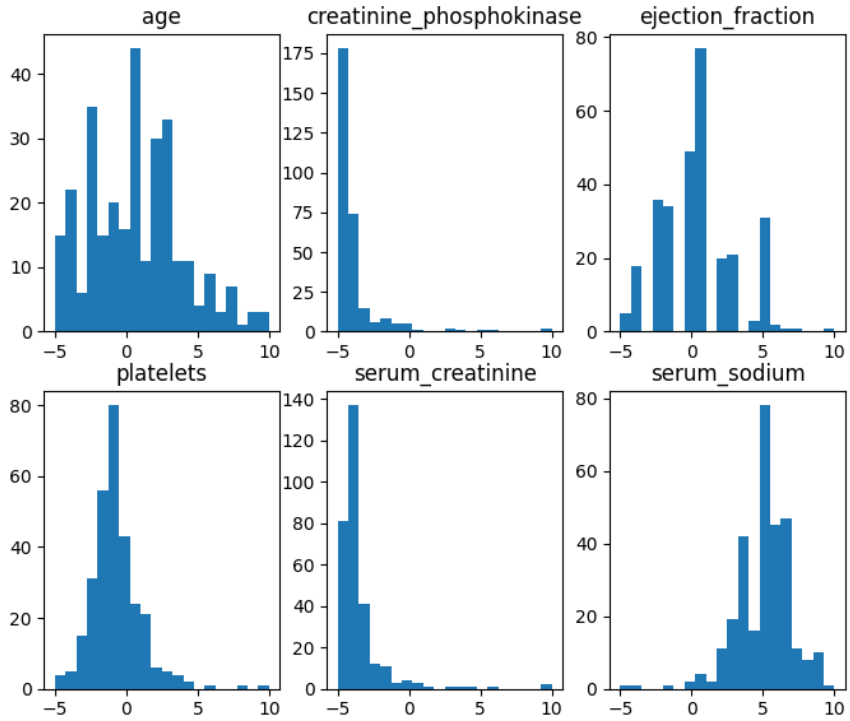

Рисунок 7 – Гистограммы распределений признаков после приведения данных к диапазону [-5;10]

Нелинейные преобразования

Приведем данные к равномерному распределению используя *QuantileTransformer.* На рисунке 8 представлены гистограммы распределений признаков после приведения их к равномерному распределению.

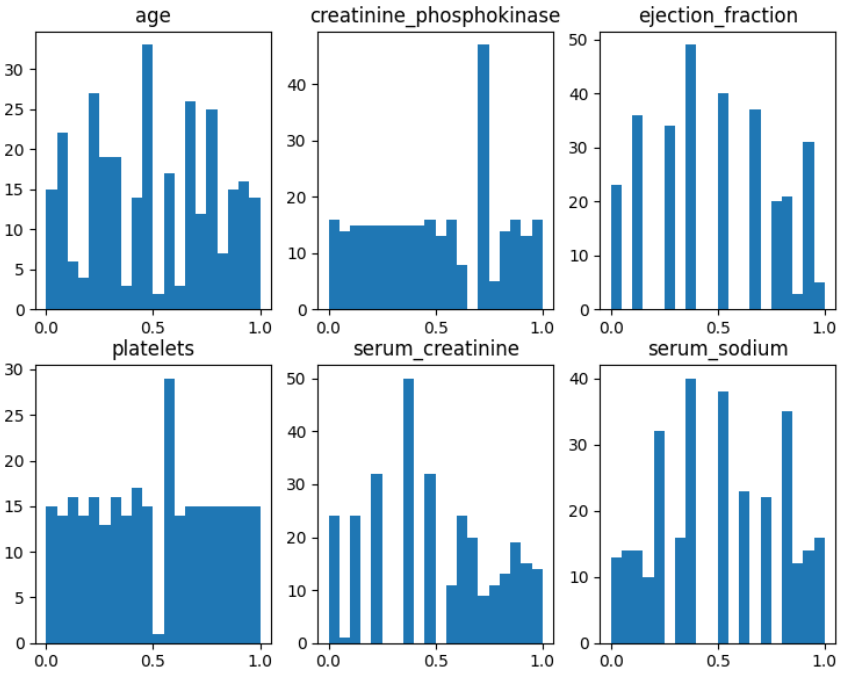

Рисунок 8 – Гистограммы распределений признаков после приведения данных к равномерному распределению при помощи *QuantileTransformer*

Как можно увидеть из рисунка 8, распределения данных стало действительно схожим с равномерным на промежутке [0;1]. Параметр *n_quantiles* объекта *QuantileTransformer* определяет количество квантилей, на которые будут разбиваться данные при построении эмпирической функции распределения вероятностей. Таким образом большее значение *n_quantiles* обеспечивает более точное преобразование данных.

Приведем данные к нормальному распределению используя *QuantileTransformer*. На рисунке 9 представлены гистограммы распределений признаков после приведения их к нормальному распределению.

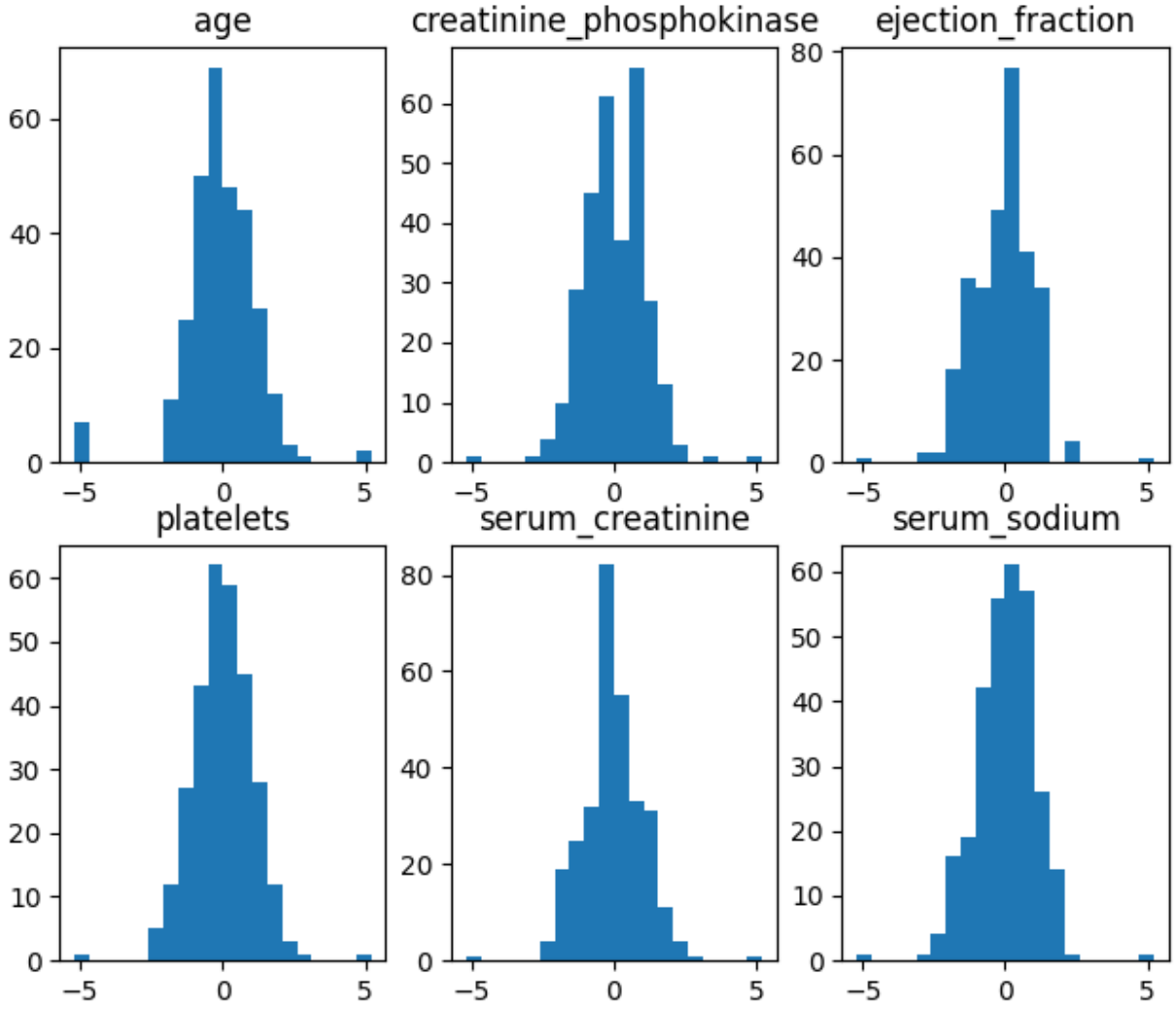

Рисунок 9 – Гистограммы распределений признаков после приведения данных к нормальному распределению при помощи *QuantileTransformer*

Как можно увидеть из рисунка 9, данные были приведены к центрированному и нормированному нормальному распределению. Приведем

данные к нормальному распределению используя *PowerTransformer*. На рисунке 10 представлены гистограммы распределений признаков после приведения их к нормальному распределению.

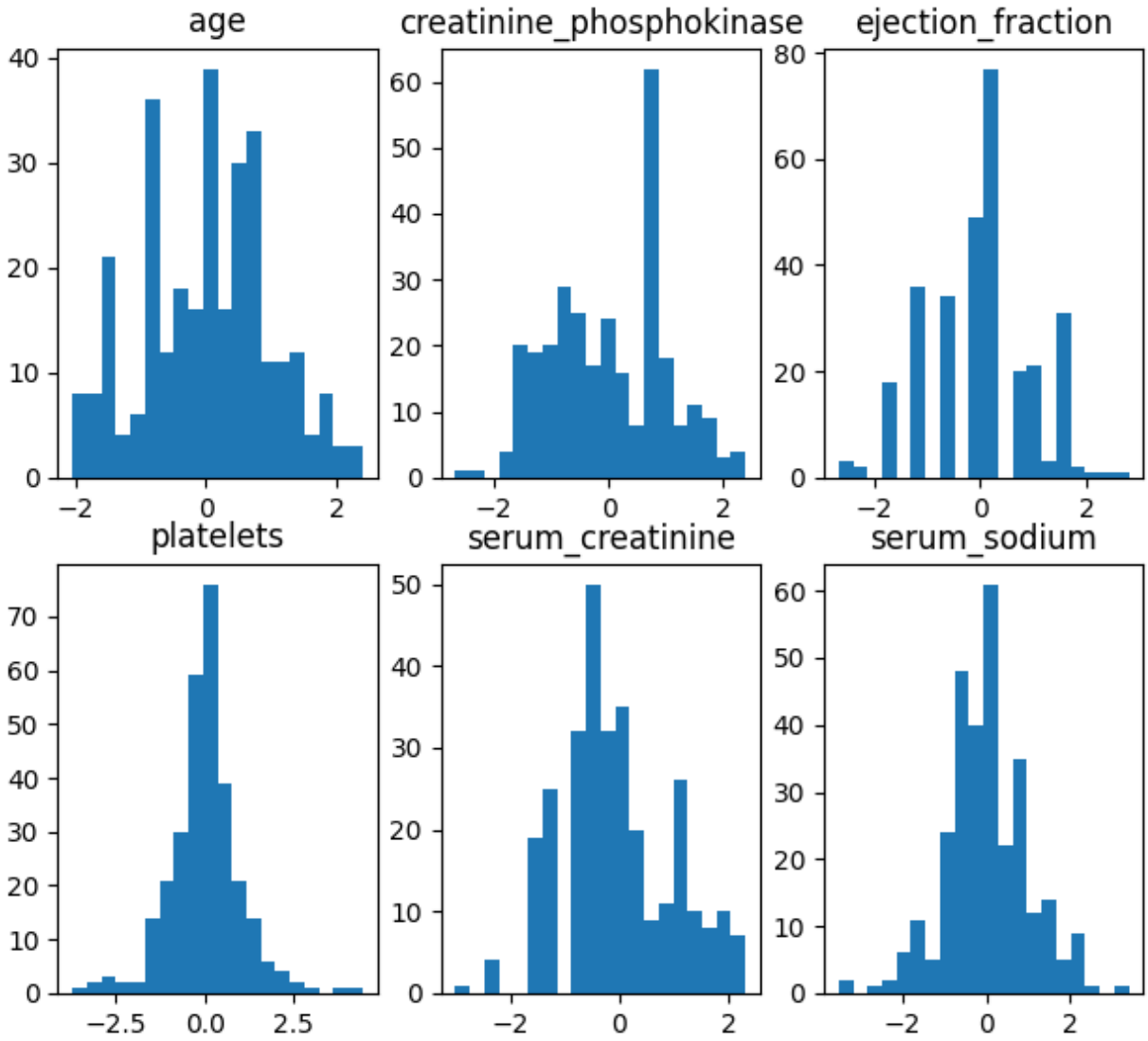

Рисунок 10 – Гистограммы распределений признаков после приведения данных к нормальному распределению при помощи *PowerTransformer*

PowerTransformer также преобразовал данные к центрированному и нормированному нормальному распределению.

Дискретизация признаков

Проведем дискретизацию признаков, используя *KBinsDiscretizer*. Для признака *age* зададим количество диапазонов, равное 3, для *creatinine_phosphokinase* – 4, для *ejection_fraction* – 3, для *platelets* – 4, для

serum_creatinine – 2, для *serum_sodium* – 4. На рисунке 11 представлены гистограммы распределений признаков после дискретизации.

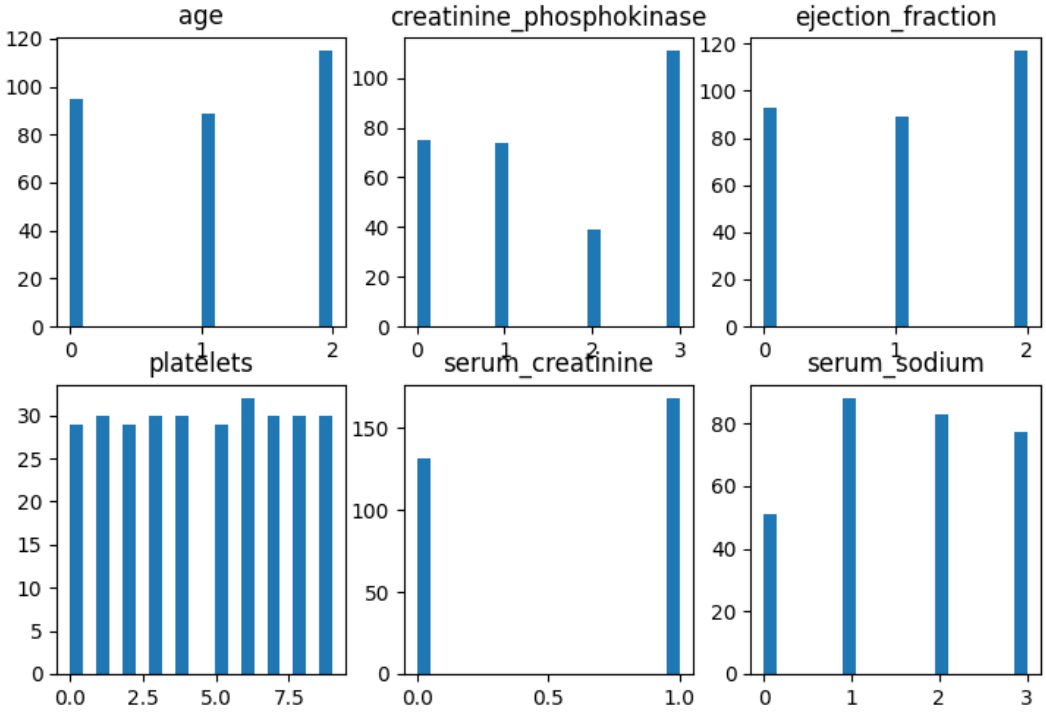

Рисунок 11 – Гистограммы распределений признаков после дискретизации при помощи *KBinsDiscretizer*

Как можно увидеть из гистограмм, приведённых на рисунке 10, после дискретизации, данные всех признаков поделились на заданное количество диапазонов. Воспользуемся признаком *bin_edges_* объекта *KBinsDiscretizer* для просмотра интервалов, которые были использованы для дискретизации. В таблице 6 представлены интервалы дискретизации.

Таблица 6 – Интервалы дискретизации признаков

Признак	Интервалы
age	[40;55] [55;65] [65;95]
creatinine phosphokinase	[23;116.5] [116.5;250] [250;582] [582;7861]
ejection fraction	[14;35] [35;40] [40;80]
platelets	[25100;153000] [153000;196000] [196000;221000] [221000;237000] [237000;262000] [262000;265000] [265000;285200] [285200;319800] [319800;374600] [374600;850000]

Продолжение таблицы 6

serum creatinine	[0.5;1.1] [1.1;9.4]
serum sodium	[113;134] [134;137] [137;140] [140;148]

Выводы

В ходе выполнения лабораторной работы было проведено ознакомление с методами предобработки данных из библиотеки Scikit Learn.

Была проведена стандартизация данных при помощи *StandardScaler*. Было выяснено, что при помощи стандартизации данные можно центрировать, сделав из математическое ожидание равным 0 и нормировать, сделав из среднеквадратичное отклонение равным 1.

Было проведено приведение данных к диапазону различными методами. *MinMaxScaler* приводит данные к диапазону [0;1], *MaxAbsScaler* делит данные на их максимальное по модулю значение, а *RobustScaler* вычитает из данных их медиану, после чего делит на величину межквартильного размаха. Также была написана функция для приведения данных к диапазону [-5;10].

Были проведены нелинейные преобразования данных. *QuantileTransformer* преобразует данные так, чтобы из распределения стало подчиняться какому-либо закону распределения. Также приведение данных к нормальному закону может быть осуществлено при помощи *PowerTransformer.*

Была проведена дискретизация данных, в ходе которой данные были поделены на заданное количество диапазонов и распределены по ним.

ПРИЛОЖЕНИЕ А

Код программы

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing

def plotData(data,n_bins=20):
    fig, axs = plt.subplots(2,3)
    axs[0, 0].hist(data[:,0], bins = n_bins)
    axs[0, 0].set_title('age')
    axs[0, 1].hist(data[:,1], bins = n_bins)
    axs[0, 1].set_title('creatinine_phosphokinase')
    axs[0, 2].hist(data[:,2], bins = n_bins)
    axs[0, 2].set_title('ejection_fraction')
    axs[1, 0].hist(data[:,3], bins = n_bins)
    axs[1, 0].set_title('platelets')
    axs[1, 1].hist(data[:,4], bins = n_bins)
    axs[1, 1].set_title('serum_creatinine')
    axs[1, 2].hist(data[:,5], bins = n_bins)
    axs[1, 2].set_title('serum_sodium')
    plt.show()

def printDataParams(data):
    dtr=data.transpose()
    print("Min:",end=' ')
    for d in dtr:
        print("%0.2f" % np.min(d), end='\t')
    print()
    print("Max:",end=' ')
    for d in dtr:
        print("%0.2f" % np.max(d), end='\t')
    print()
    print("Mean:",end=' ')
    for d in dtr:
        print("%0.2f" % np.mean(d), end='\t')
    print()
    print("STD: ",end=' ')
    for d in dtr:
        print("%0.2f" % np.std(d), end='\t')
    print("\n")

def dataToMinus5toPlus10(data):
    min_max_scaler = preprocessing.MinMaxScaler()
    data_min_max_scaled = min_max_scaler.fit_transform(data)
    data_min_max_scaled*=15
    data_min_max_scaled-=5
    return data_min_max_scaled

df = pd.read_csv('heart_failure_clinical_records_dataset.csv')
df                                  =                                df.drop(columns
=['anaemia','diabetes','high_blood_pressure','sex','smoking','time','DEATH_EV
ENT'])
print(df)
data = df.to_numpy(dtype='float')
print("Before standartization:")
plotData(data)
printDataParams(data)

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

```
print("After standartization based on the first 150 observations:")
plotData(data_scaled)
printDataParams(data_scaled)

print("Params of scaler:")
print("mean_:",scaler.mean_)
print("scale_:",scaler.scale_)
print()

scaler = preprocessing.StandardScaler().fit(data[:,:])
data_scaled = scaler.transform(data)
print("After standartization based on all data:")
plotData(data_scaled)
printDataParams(data_scaled)

min_max_scaler = preprocessing.MinMaxScaler()
data_min_max_scaled = min_max_scaler.fit_transform(data)
plotData(data_min_max_scaled)
print("min_max_scaler data_min_:",min_max_scaler.data_min_)
print("min_max_scaler data_max_:",min_max_scaler.data_max_)
print()

max_abs_scaler = preprocessing.MaxAbsScaler()
data_max_abs_scaled = max_abs_scaler.fit_transform(data)
plotData(data_max_abs_scaled)

robust_scaler = preprocessing.RobustScaler()
data_robust_scaled = robust_scaler.fit_transform(data)
plotData(data_robust_scaled)

data_Minus5toPlus10=dataToMinus5toPlus10(data)
plotData(data_Minus5toPlus10)

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

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

power_transformer = preprocessing.PowerTransformer()
data_power_transformed = power_transformer.fit_transform(data)
plotData(data_power_transformed)

k_bins_discreaser=preprocessing.KBinsDiscretizer(n_bins=[3,   4,   3,   10   ,2
,4],encode='ordinal')
data_k_bins_discreased=k_bins_discreaser.fit_transform(data)
print(data_k_bins_discreased)
plotData(data_k_bins_discreased)
for atr in k_bins_discreaser.bin_edges_:
    lft=None
    for rgh in atr:
        if lft:
            print('[%.2f;%.2f]'%(lft,rgh),end=' ')
        lft=rgh
    print()
```