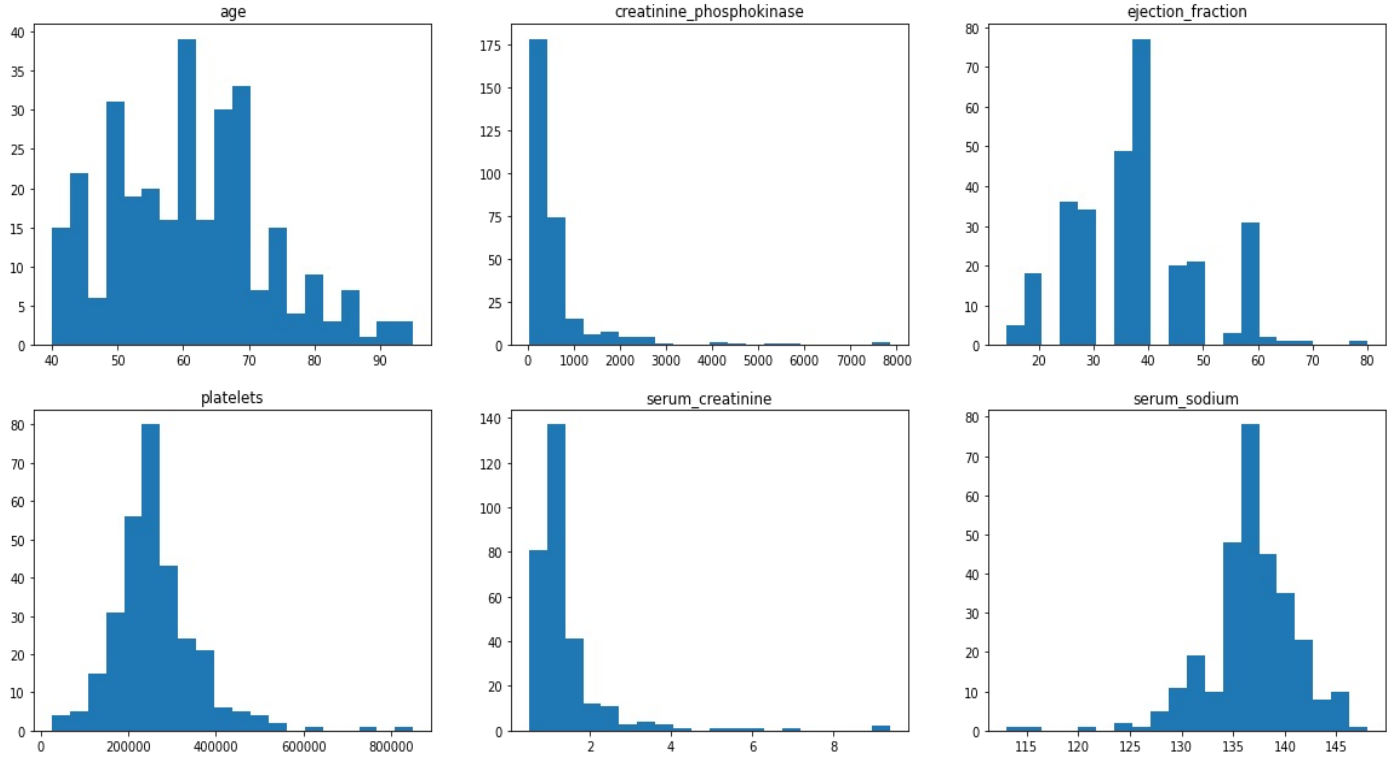
Лабораторная №1

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','time','DEATH_EVENT'])

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


      age  creatinine_phosphokinase  ejection_fraction  platelets  \
0    75.0                       582                 20  265000.00
1    55.0                      7861                 38  263358.03
2    65.0                       146                 20  162000.00
3    50.0                       111                 20  210000.00
4    65.0                       160                 20  327000.00
..    ...                       ...                ...        ...
294  62.0                        61                 38  155000.00
295  55.0                      1820                 38  270000.00
296  45.0                      2060                 60  742000.00
297  45.0                      2413                 38  140000.00
298  50.0                       196                 45  395000.00

     serum_creatinine  serum_sodium
0                 1.9           130
1                 1.1           136
2                 1.3           129
3                 1.9           137
4                 2.7           116
..                ...           ...
294               1.1           143
295               1.2           139
296               0.8           138
297               1.4           140
298               1.6           136

[299 rows x 6 columns]


import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = (20, 10)

n_bins = 20

fig, axs = plt.subplots(2,3)

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')

plt.show()
```



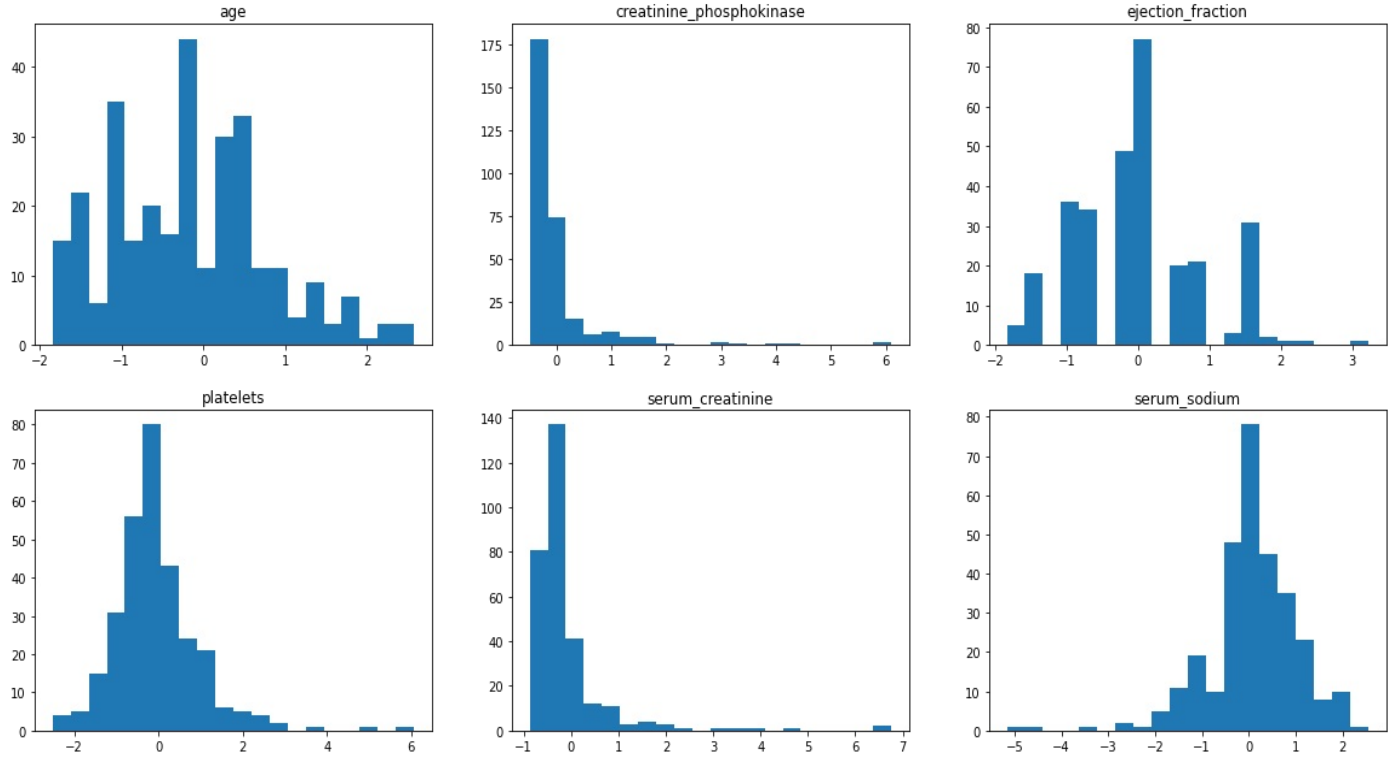
Значения по гистограммам:

	Age	CP	EF	Platelets	SC	Serim Sodium
Приблизительный Диапазон	[40, 96]	[0, 8000]	[0 , 80]	[0, 800000]	[0, 9]	[0, 150]
Значение, близкое к среднему	60	200	38	220000	1	136

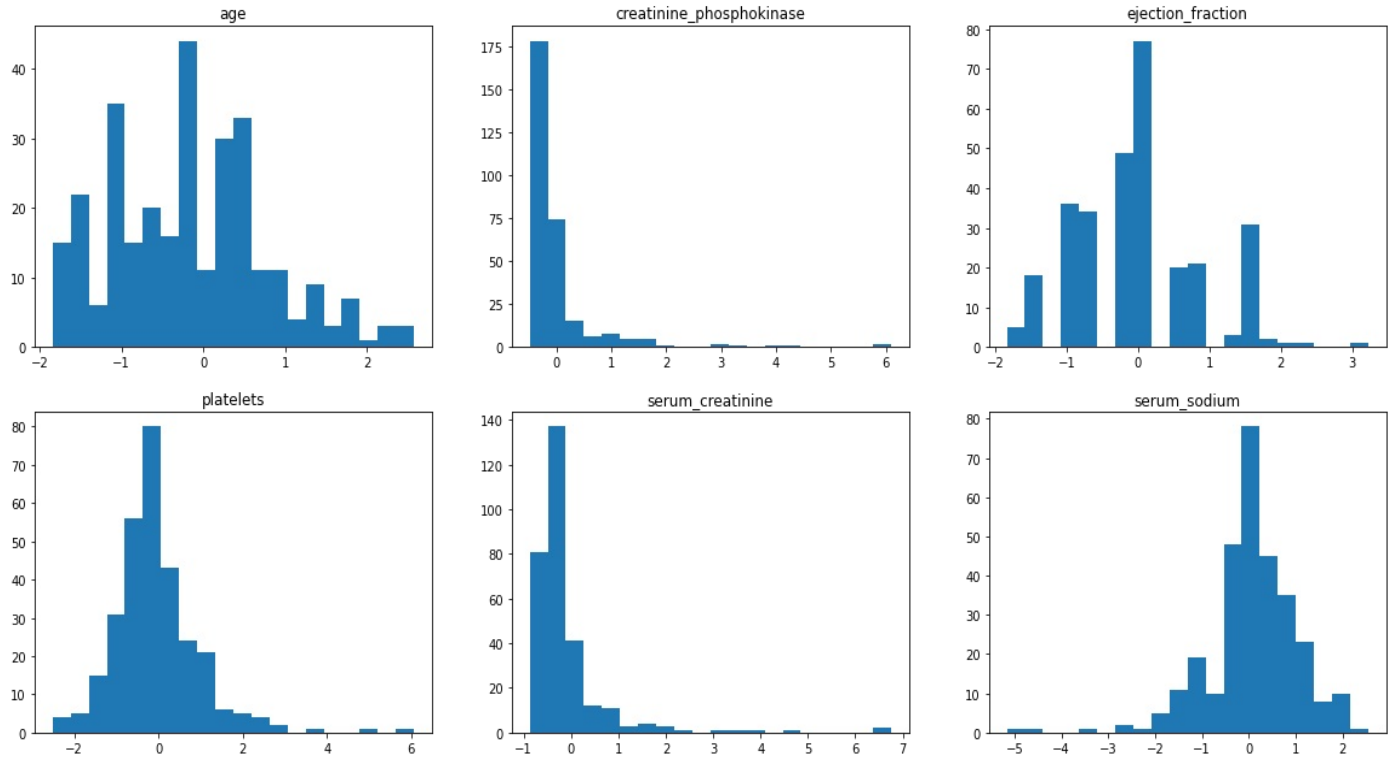
```
data = df.to_numpy(dtype='float')
```

Стандартизация данных

```
from sklearn import preprocessing

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


data_scaled = scaler.transform(data)


fig, axs = plt.subplots(2,3)

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')

plt.show()
```




```
fig, axs = plt.subplots(2,3)

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')

plt.show()
```



Что изменилось и почему?

Изменились мат ожидания на 0, а также дисперсии стали сравнимы. Это произошло потому что у кажжого значения набора отняли мат ожидание а и разделили его на СКО. Нормализация произведена с целью, чтобы статистически сравнить величины разных размерностей и порядков.

Расчет параметров:

```
mean_age = data[:, 0].mean()
std_age = data[:, 0].std()
mean_cp = data[:, 1].mean()
std_cp = data[:, 1].std()
mean_ef = data[:, 2].mean()
std_ef = data[:, 2].std()
mean_plat = data[:, 3].mean()
std_plat = data[:, 3].std()
mean_sc = data[:, 4].mean()
std_sc = data[:, 4].std()
mean_ss = data[:, 5].mean()
std_ss = data[:, 5].std()

print(f'Age mean (no scaling): {mean_age}')
print(f'Age std (no scaling): {std_age}')
print(f'Cp mean (no scaling): {mean_cp}')
print(f'Cp std (no scaling): {std_cp}')
print(f'Ef mean (no scaling): {mean_ef}')
print(f'Ef std (no scaling): {std_ef}')
print(f'Plat mean (no scaling): {mean_plat}')
print(f'Plat std (no scaling): {std_plat}')
print(f'Sc mean (no scaling): {mean_sc}')
print(f'Sc std (no scaling): {std_sc}')
print(f'Ss mean (no scaling): {mean_ss}')
print(f'Ss std (no scaling): {std_ss}')

mean_age = data_scaled[:, 0].mean()
std_age = data_scaled[:, 0].std()
mean_cp = data_scaled[:, 1].mean()
std_cp = data_scaled[:, 1].std()
mean_ef = data_scaled[:, 2].mean()
std_ef = data_scaled[:, 2].std()
mean_plat = data_scaled[:, 3].mean()
std_plat = data_scaled[:, 3].std()
mean_sc = data_scaled[:, 4].mean()
std_sc = data_scaled[:, 4].std()
```

```
mean_ss = data_scaled[:, 5].mean()
std_ss = data_scaled[:, 5].std()

print(f'Age mean: {mean_age}')
print(f'Age std: {std_age}')
print(f'Cp mean: {mean_cp}')
print(f'Cp std: {std_cp}')
print(f'Ef mean: {mean_ef}')
print(f'Ef std: {std_ef}')
print(f'Plat mean: {mean_plat}')
print(f'Plat std: {std_plat}')
print(f'Sc mean: {mean_sc}')
print(f'Sc std: {std_sc}')
print(f'Ss mean: {mean_ss}')
print(f'Ss std: {std_ss}')
```

```
Age mean (no scaling): 60.83389297658862
Age std (no scaling): 11.874901429842655
Cp mean (no scaling): 581.8394648829432
Cp std (no scaling): 968.6639668032415
Ef mean (no scaling): 38.08361204013378
Ef std (no scaling): 11.815033462318585
Plat mean (no scaling): 263358.02926421404
Plat std (no scaling): 97640.54765451424
Sc mean (no scaling): 1.3938795986622072
Sc std (no scaling): 1.0327786652795918
Ss mean (no scaling): 136.62541806020067
Ss std (no scaling): 4.405092379513557
Age mean: -0.16970362369106984
Age std: 0.9538237876978354
Cp mean: -0.021276750290383013
Cp std: 0.8141790488228113
Ef mean: 0.01050249484809085
Ef std: 0.9061082161919123
Plat mean: -0.035228788194085287
Plat std: 1.0150611342848024
Sc mean: -0.10864080163893569
Sc std: 0.8854288727548568
Ss mean: 0.03790759894920013
Ss std: 0.9703735961735016
```

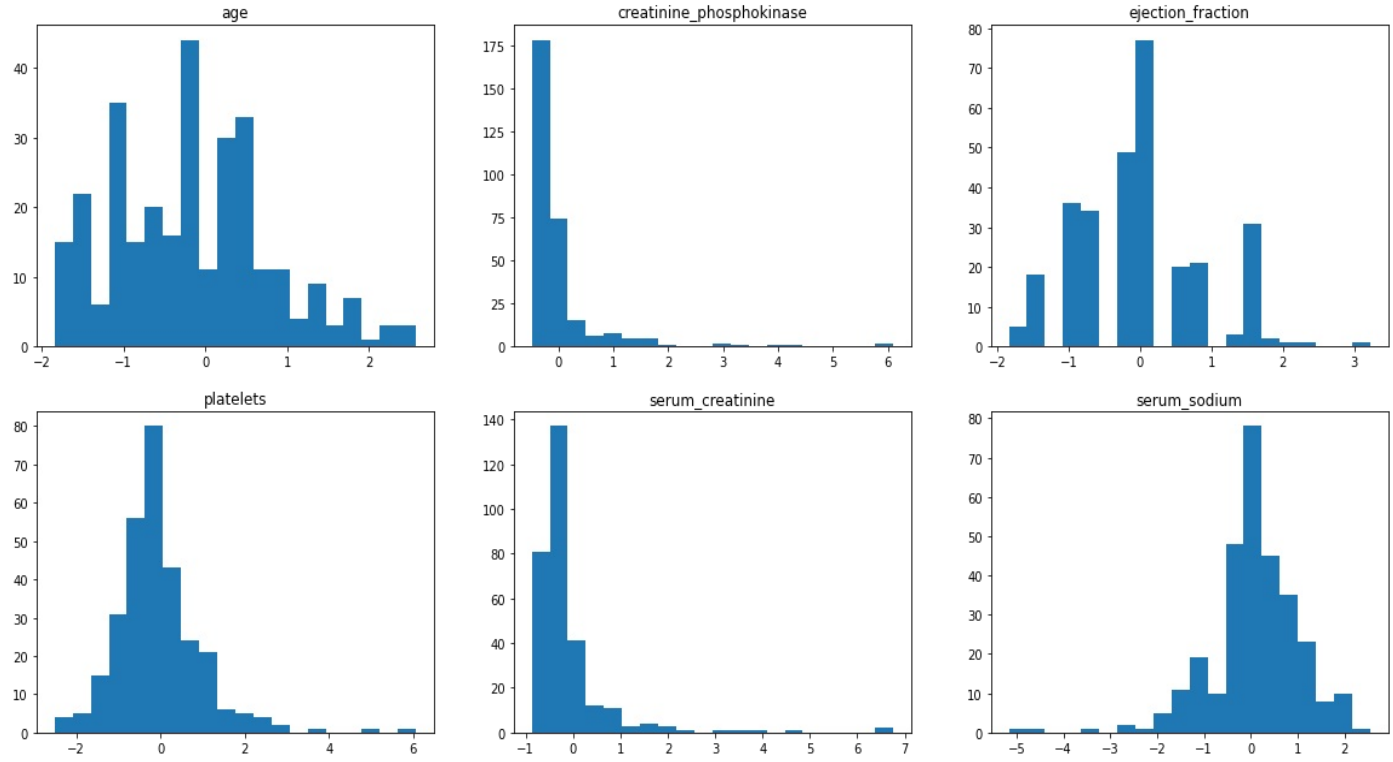
```
import math

mean_age = scaler.mean_[0]
std_age = math.sqrt(scaler.var_[0])
mean_cp = scaler.mean_[1]
std_cp = math.sqrt(scaler.var_[1])
mean_ef = scaler.mean_[2]
std_ef = math.sqrt(scaler.var_[2])
mean_plat = scaler.mean_[3]
std_plat = math.sqrt(scaler.var_[3])
mean_sc = scaler.mean_[4]
std_sc = math.sqrt(scaler.var_[4])
mean_ss = scaler.mean_[5]
std_ss = math.sqrt(scaler.var_[5])

print(f'Age mean: {mean_age}')
print(f'Age std: {std_age}')
print(f'Cp mean: {mean_cp}')
print(f'Cp std: {std_cp}')
print(f'Ef mean: {mean_ef}')
print(f'Ef std: {std_ef}')
print(f'Plat mean: {mean_plat}')
print(f'Plat std: {std_plat}')
print(f'Sc mean: {mean_sc}')
print(f'Sc std: {std_sc}')
print(f'Ss mean: {mean_ss}')
print(f'Ss std: {std_ss}')
```

```
Age mean: 62.946666666666665
Age std: 12.449785361826748
Cp mean: 607.1533333333333
Cp std: 1189.7431752926996
Ef mean: 37.946666666666665
Ef std: 13.039318318923817
Plat mean: 266746.74946666666
Plat std: 96191.79018543586
Sc mean: 1.5206000000000002
Sc std: 1.1664162950393542
Ss mean: 136.45333333333335
Ss std: 4.539583926112858
```

Формула нормализации: $$ z = {{x - u} \over {\sigma}} $$

где x - случайна величина, u - среднее, σ - СКО

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


fig, axs = plt.subplots(2,3)

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')

plt.show()
```

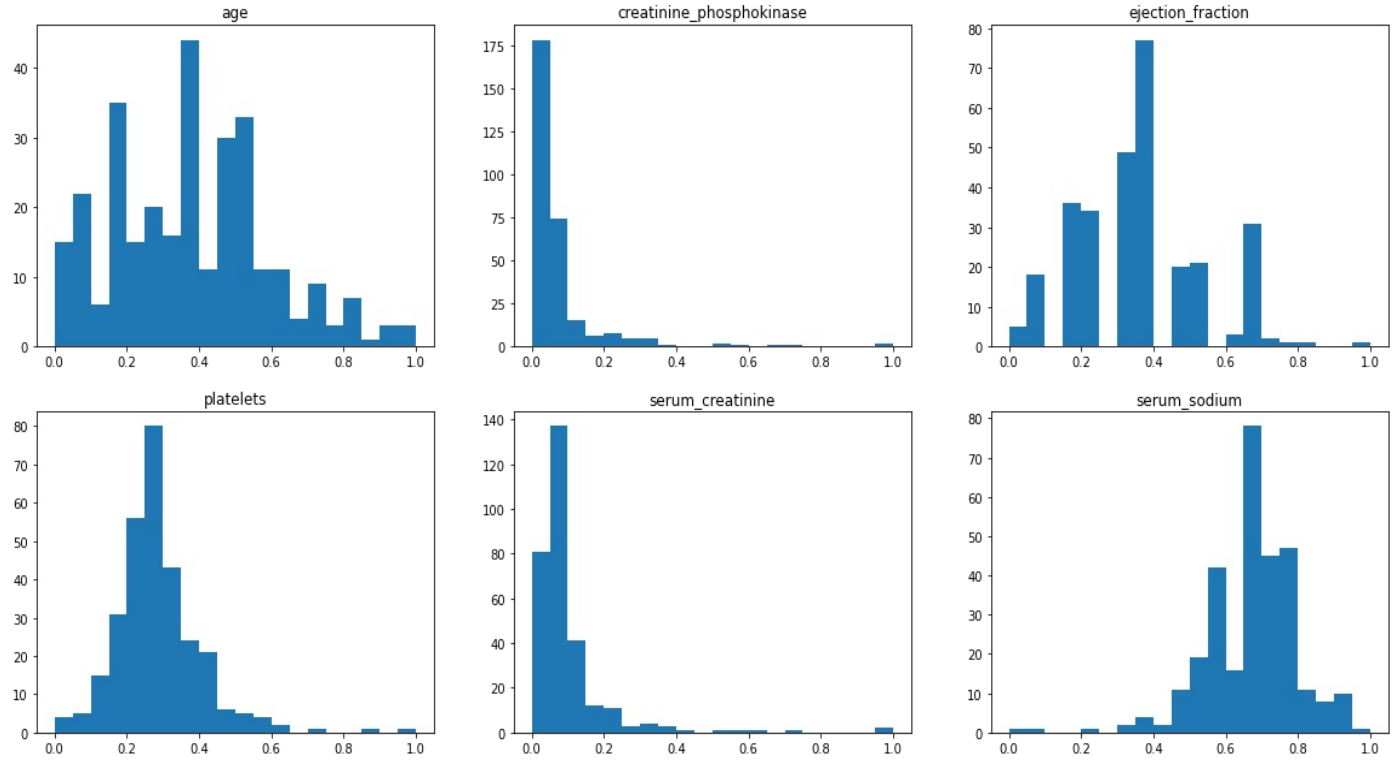


```
import math

mean_age = scaler.mean_[0]
std_age = math.sqrt(scaler.var_[0])
mean_cp = scaler.mean_[1]
std_cp = math.sqrt(scaler.var_[1])
mean_ef = scaler.mean_[2]
std_ef = math.sqrt(scaler.var_[2])
mean_plat = scaler.mean_[3]
std_plat = math.sqrt(scaler.var_[3])
mean_sc = scaler.mean_[4]
std_sc = math.sqrt(scaler.var_[4])
mean_ss = scaler.mean_[5]
std_ss = math.sqrt(scaler.var_[5])

print(f'Age mean: {mean_age}')
print(f'Age std: {std_age}')
print(f'Cp mean: {mean_cp}')
print(f'Cp std: {std_cp}')
print(f'Ef mean: {mean_ef}')
print(f'Ef std: {std_ef}')
print(f'Plat mean: {mean_plat}')
print(f'Plat std: {std_plat}')
print(f'Sc mean: {mean_sc}')
print(f'Sc std: {std_sc}')
print(f'Ss mean: {mean_ss}')
print(f'Ss std: {std_ss}')


Age mean: 62.946666666666665
Age std: 12.449785361826748
Cp mean: 607.1533333333333
Cp std: 1189.7431752926996
Ef mean: 37.946666666666665
Ef std: 13.039318318923817
Plat mean: 266746.74946666666
Plat std: 96191.79018543586
Sc mean: 1.5206000000000002
Sc std: 1.1664162950393542
Ss mean: 136.45333333333335
Ss std: 4.539583926112858
```

Приведение к диапазону

MinMaxScaler

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


fig, axs = plt.subplots(2,3)

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')

plt.show()
```

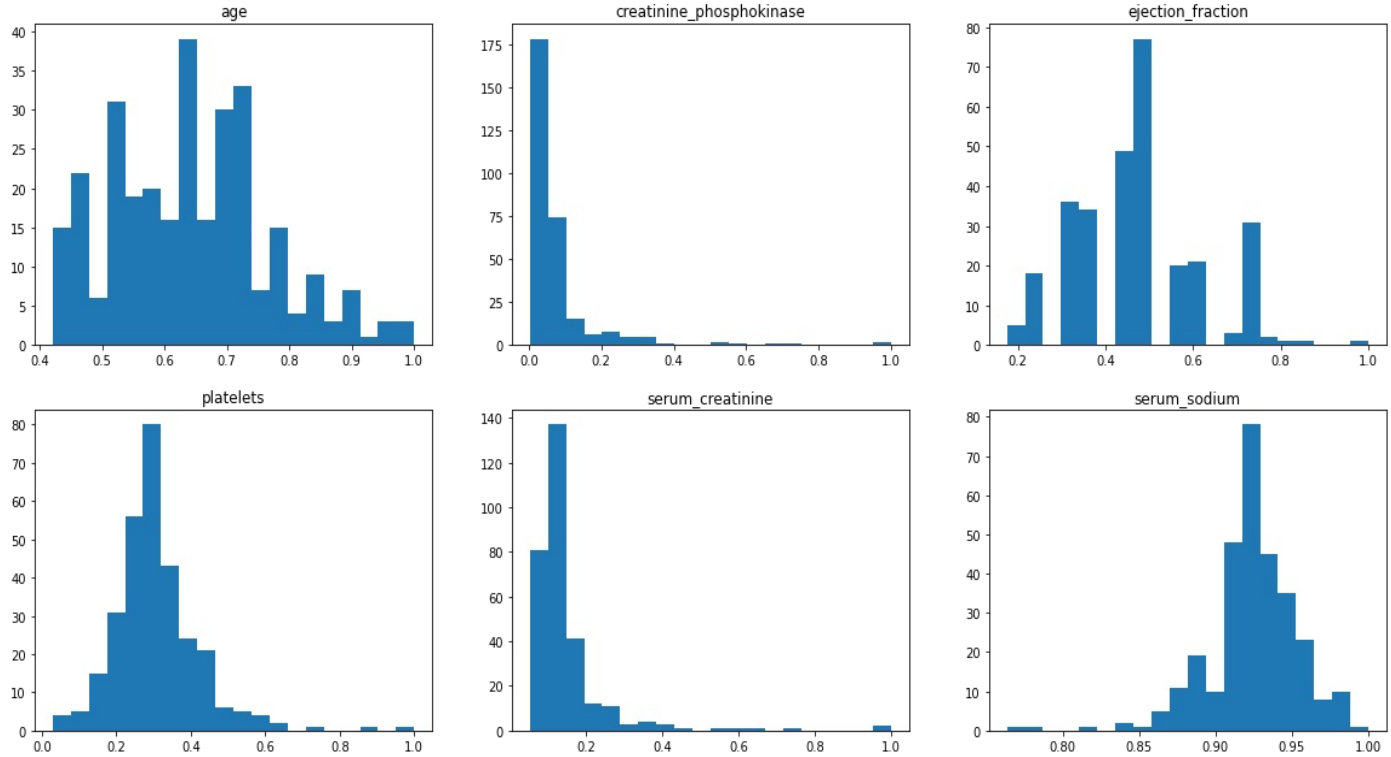



```
print(min_max_scaler.data_min_[0])
print(min_max_scaler.data_max_[0])
print('\n')
print(min_max_scaler.data_min_[1])
print(min_max_scaler.data_max_[1])
print('\n')
print(min_max_scaler.data_min_[2])
print(min_max_scaler.data_max_[2])
print('\n')
print(min_max_scaler.data_min_[3])
print(min_max_scaler.data_max_[3])
print('\n')
print(min_max_scaler.data_min_[4])
print(min_max_scaler.data_max_[4])
print('\n')
print(min_max_scaler.data_min_[5])
print(min_max_scaler.data_max_[5])
```

```
40.0
95.0


23.0
7861.0


14.0
80.0


25100.0
850000.0


0.5
9.4


113.0
148.0
```

MaxAbsScaler

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


fig, axs = plt.subplots(2,3)

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')

plt.show()
```

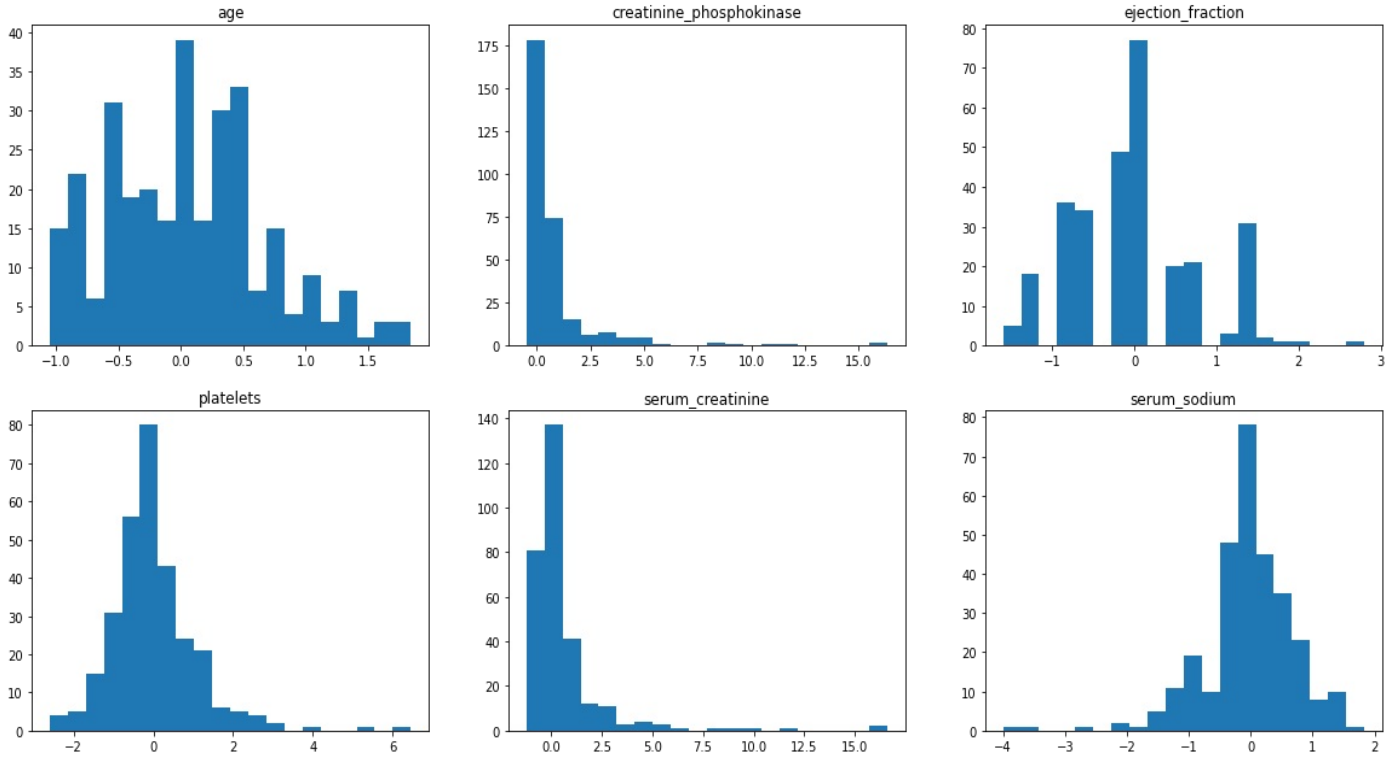


MaxAbsScaler делит данные на максимальное обсолютное значение, при это разброс не меняется.

RobustScaler

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


fig, axs = plt.subplots(2,3)

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')

plt.show()
```

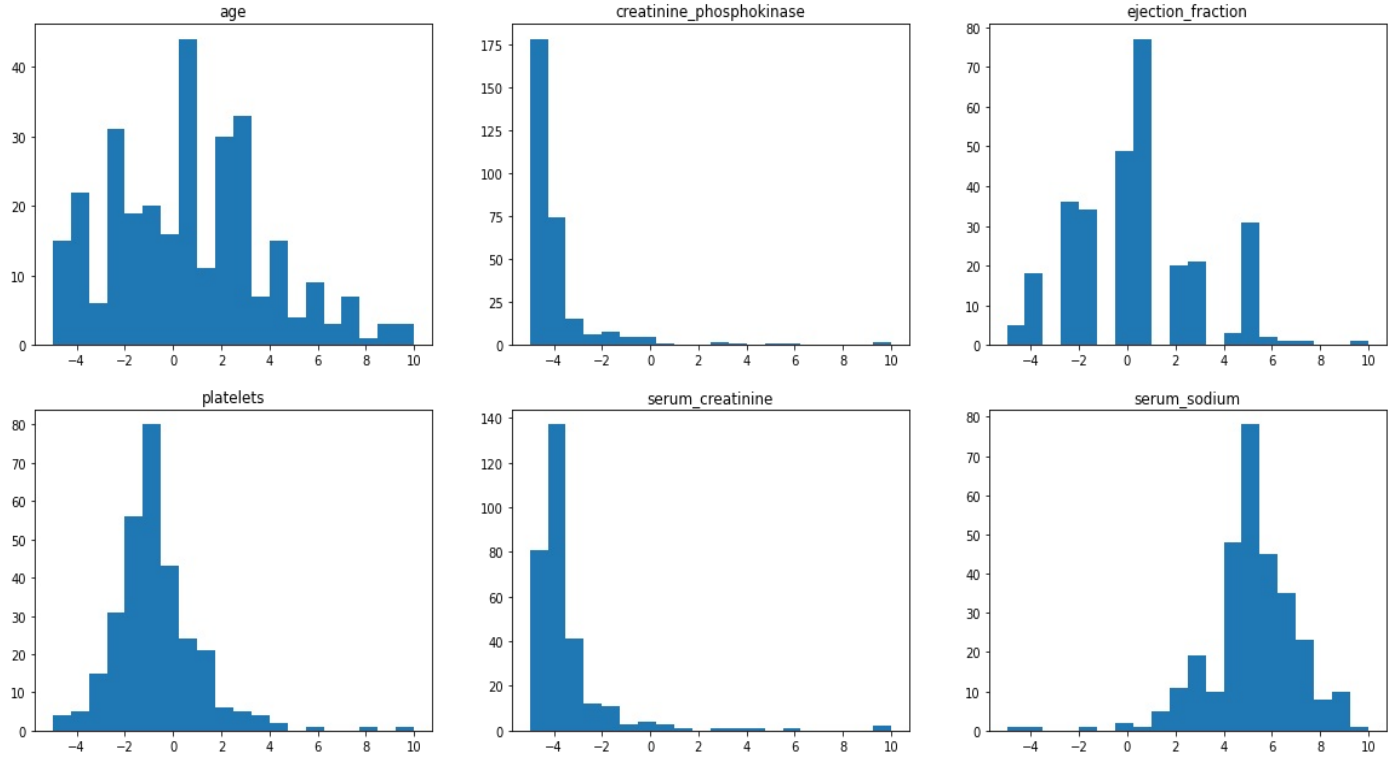


`RobustScaler` похож на `StandardScaler`, но для стандартизации использует не среднее и СКО, а медиану и IQR (Intequartile range, расстояние между первым и третьи квартилем, т.е. “ширину места, вероятность в которое попасть 50%”). Этот метод прведения более устойчив к выбросам, чем `StandardScaler`, поэтому так и называется.

Приведение к промежутку [-5, 10]

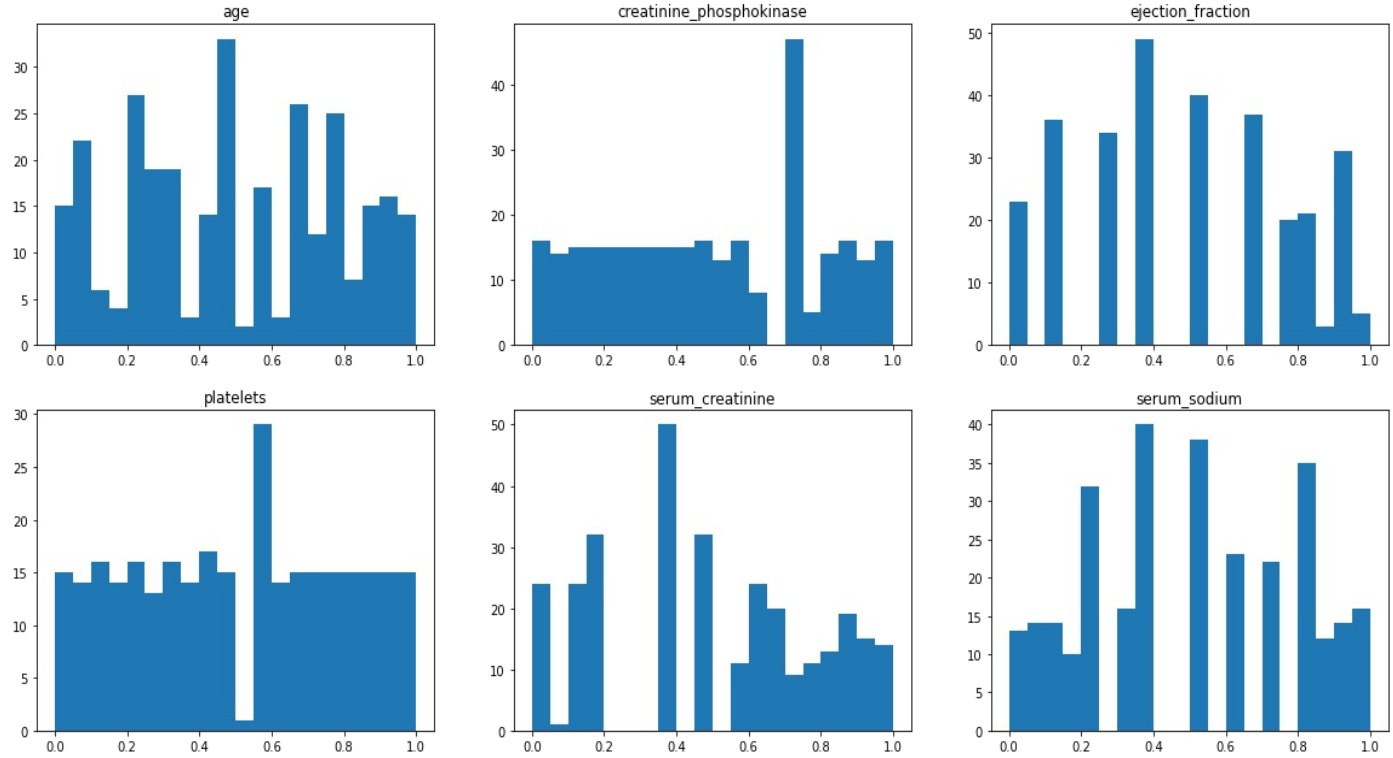
```
min_max_scaler = preprocessing.MinMaxScaler((-5, 10)).fit(data)
data_min_max_scaled = min_max_scaler.transform(data)


fig, axs = plt.subplots(2,3)

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')
```

```
plt.show()
```

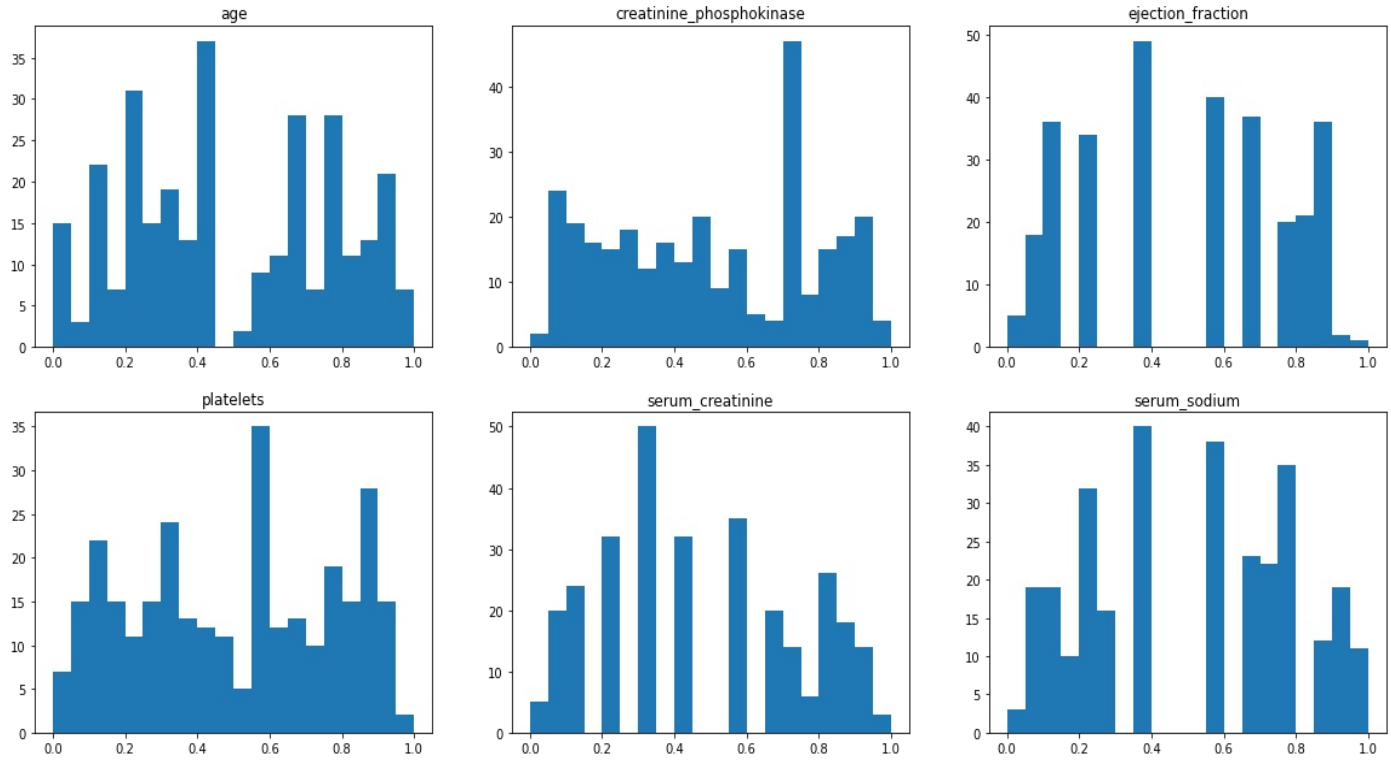


Нелинейные преобразования

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


fig, axs = plt.subplots(2,3)

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')

plt.show()
```



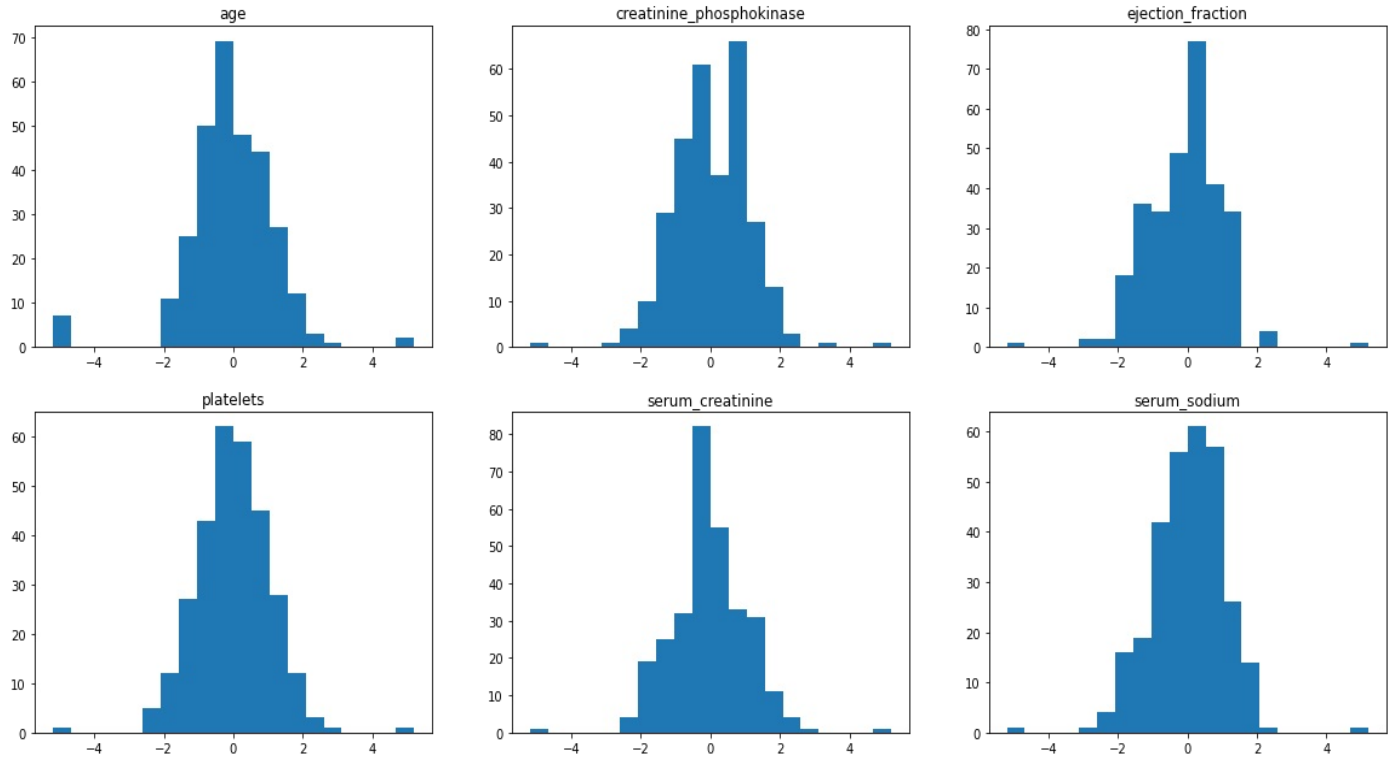

```
quantile_transformer = preprocessing.QuantileTransformer(n_quantiles = 10,random_state=0).fit(data)
data_quantile_scaled = quantile_transformer.transform(data)


fig, axs = plt.subplots(2,3)

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')

plt.show()
```



`n_quantiles` влияет на то, насколько хорошо приведется исходные данные к равномерному распредлению (в данным случае). Это значение используется для задания размера шага при дискретизации оценочной CDF для исходных данных (CDF используется для приведения к равномерному распределению в `QuantileTransformer`).

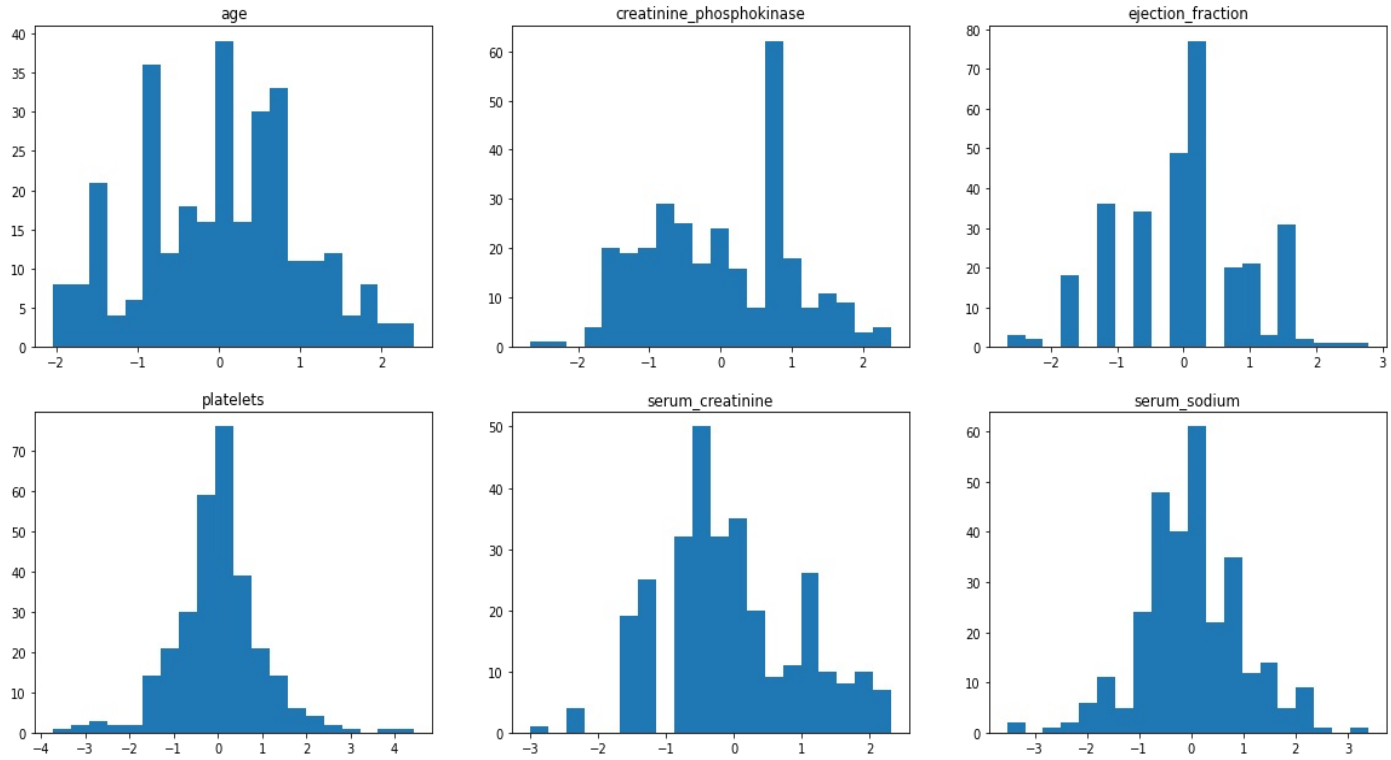
Приведение к номарльному распределению

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


fig, axs = plt.subplots(2,3)

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')

plt.show()
```



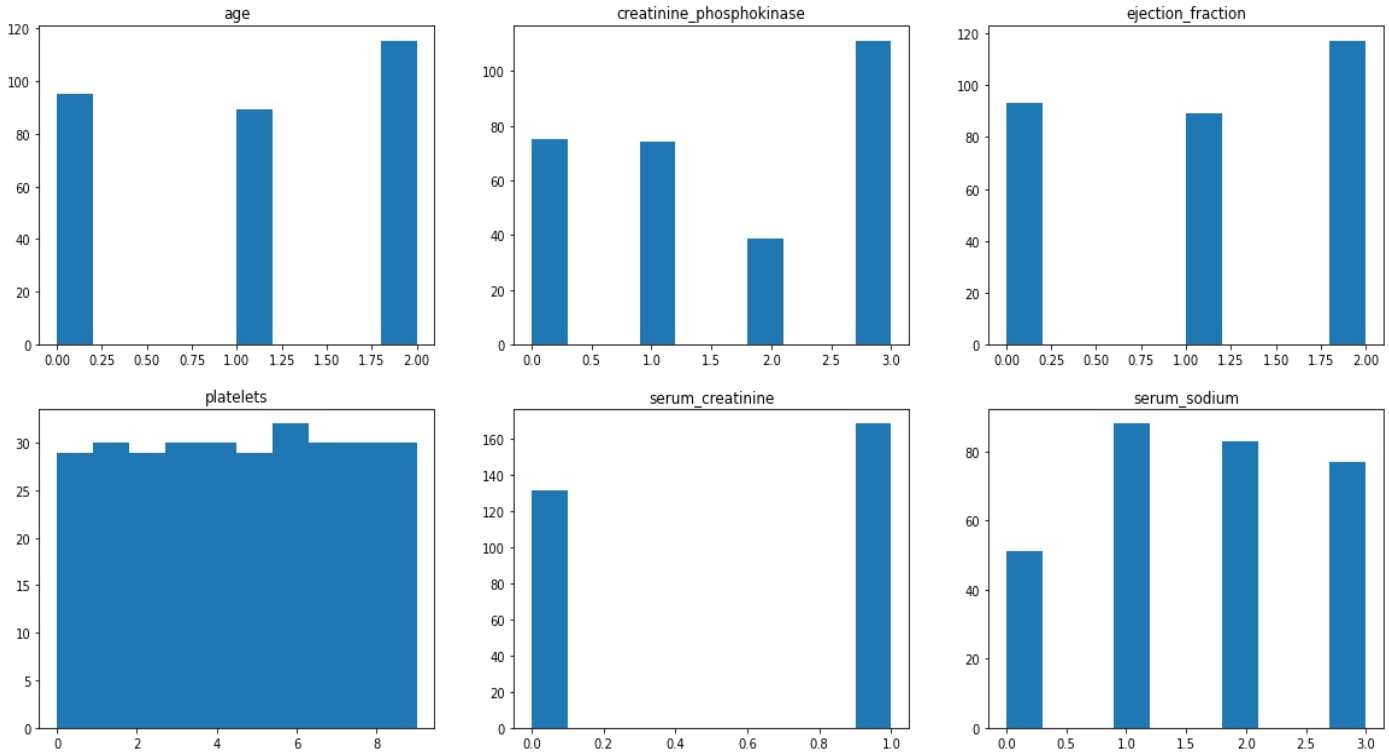
PowerTransformer

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


fig, axs = plt.subplots(2,3)

axs[0, 0].hist(data_power_scaled[:,0], bins = n_bins)c
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')

plt.show()
```



Дискретизация признаков

```
biner = preprocessing.KBinsDiscretizer(n_bins=[3, 4, 3, 10, 2, 4], encode='ordinal')
data_bined = biner.fit_transform(data)


fig, axs = plt.subplots(2,3)

axs[0, 0].hist(data_bined[:,0])
axs[0, 0].set_title('age')

axs[0, 1].hist(data_bined[:,1])
axs[0, 1].set_title('creatinine_phosphokinase')

axs[0, 2].hist(data_bined[:,2])
axs[0, 2].set_title('ejection_fraction')

axs[1, 0].hist(data_bined[:,3])
axs[1, 0].set_title('platelets')

axs[1, 1].hist(data_bined[:,4])
axs[1, 1].set_title('serum_creatinine')

axs[1, 2].hist(data_bined[:,5])
axs[1, 2].set_title('serum_sodium')

plt.show()
```



Количество столбиков = количество бинов (промежутков), на которые мы разбили данные.

```
print(biner.bin_edges_)
```

```
[array([40., 55., 65., 95.])
 array([  23. ,  116.5,  250. ,  582. , 7861. ])
 array([14., 35., 40., 80.])
 array([ 25100., 153000., 196000., 221000., 237000., 262000., 265000.,
        285200., 319800., 374600., 850000.])
 array([0.5, 1.1, 9.4]) array([113., 134., 137., 140., 148.])]
```