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

ОТЧЕТ

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

Студент гр. 8303 _____ Абибулаев Э.Э.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

1 Цель работы.

Ознакомиться с методами предобработки данных из библиотеки Scikit Learn

2 Выполнение работы.

1. Загрузка данных

1.1. Загрузжен датасет по ссылке: https://www.kaggle.com/andrewmvd/heart-failure-clinical-data. Данные представлены в виде csv таблицы.

1.2. Создан Python скрипт. Загружен датасет в датафрейм, и исключены бинарные признаки и признаки времени.

1.3. Построена гистограмма признаков (рис. 1)

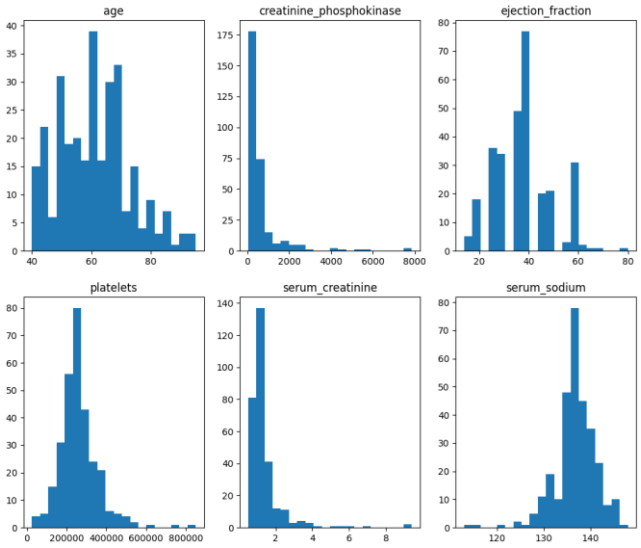

Рисунок 1: Гистограмма признаков

1.4. На основании гистограмм определены диапазоны значений для каждого из признаков, а также возле какого значения лежит наибольшее количество наблюдений.

Age — 40...95, Макс. 60

creatinine_phosphokinase — 0...8000, Макс. 0

ejection_fraction — 10…80, Макс. 38

platelets — 50 000...850 000, Макс. 240 000

serum_creatinine — 0...10, Макс. 0.5

serum_sodium — 110...150, Макс. 138

1.5. Так как библиотека Sklearn работает с NumPy массивом, то датафрейм преобразован к двумерному массиву NumPy, где строка соответствует наблюдению, а столбец признаку.

2. Стандартизация данных

2.1. Был подключен модуль Sklearn и настроена стандартизация на основе первых 150 наблюдений используя StandartScaler.

2.2. Были стандартизированы все данные.

2.3. Построены диаграммы стандартизированных данных (рис. 2)

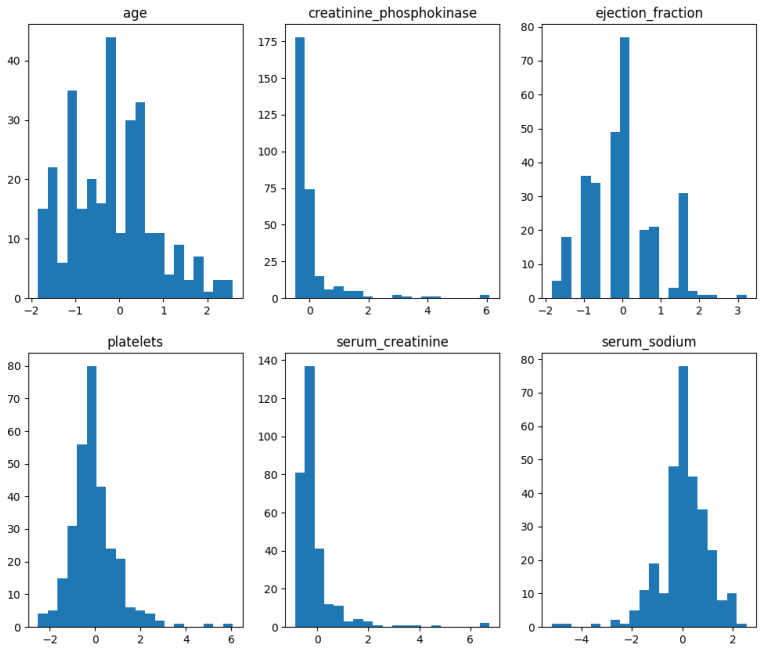

Рис. 2: Данные после стандартизации

2.4. По диаграмме видно, что данные приведены к виду, напоминающему нормальное распределение. Самый частый признак — находится в нуле, а самые редкие признаки — на максимальном отдалении.

2.5. Были рассчитаны мат. ожидание и СКО каждой выборки

Выборка	Статистика	age	Creatinine phosphokinase	Ejection fraction	platelets	Serum creatinine	Serum sodium
Оригинальная	Мат. ожидание	6.083e+01	5.818e+02	3.808e+01	2.634e+05	1.394e+00	1.366e+02
	СКО	1.187e+01	9.687e+02	1.182e+01	9.764e+04	1.033e+00	4.405e+00
Стандарти-зированная на 150	Мат. ожидание	-1.697e-01	-2.128e-02	1.050e-02	-3.523e-02	-1.086e-01	3.791e-02
	СКО	9.538e-01	8.142e-01	9.061e-01	1.015e+00	8.854e-01	9.704e-01
Стандарти-зированная	Мат. ожидание	5.703e-16	0.000e+00	-3.268e-17	7.723e-17	1.426e-16	-8.674e-16
	СКО	1.000e+00	1.000e+00	1.000e+00	1.000e+00	1.000e+00	1.000e+00

По таблице можно сделать вывод, что стандартизация производится по следующей формуле:

$$Y = \frac{X - Mean(X)}{Std(X)}$$

3. Приведение к диапазону

3.1. Данные были приведены к диапазону при помощи MinMaxScaler()

3.2. Были построены гистограммы признаков (Рис. 3)

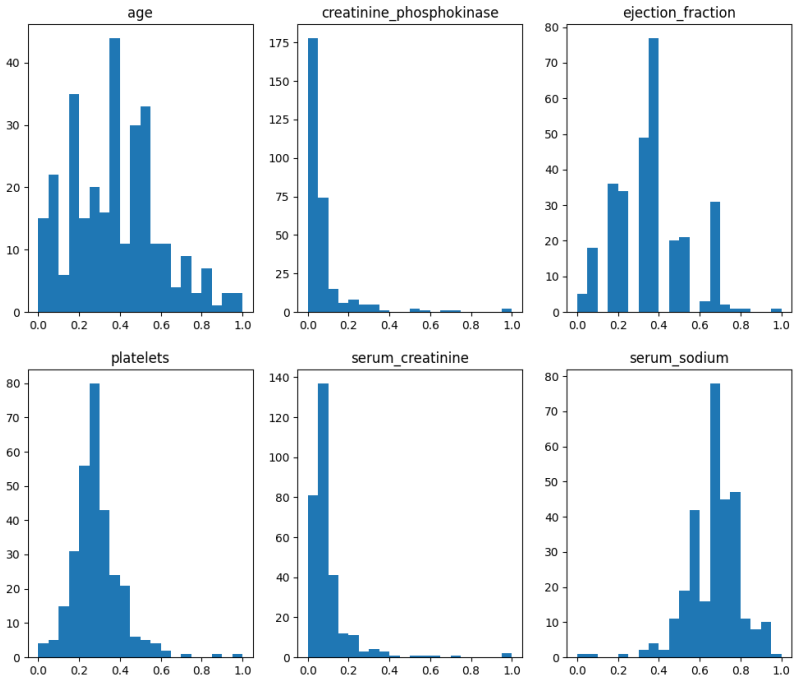

Рис. 3: Гистограмма признаков после MinMaxScaler

3.3. Были получены диапазоны изменения признаков, при помощи чтения полей min_max_scaler

Выборки	age	Creatinine phosphokinase	Ejection fraction	platelets	Serum creatinine	Serum sodium
Min	40.0	23.0	14.0	25100.0	0.5	113.0
Max	95.0	7861.0	80.0	850000.0	9.4	148.0

3.4. Так же, была произведена стандартизация при помощи MaxAbsScaler и RobustScaler

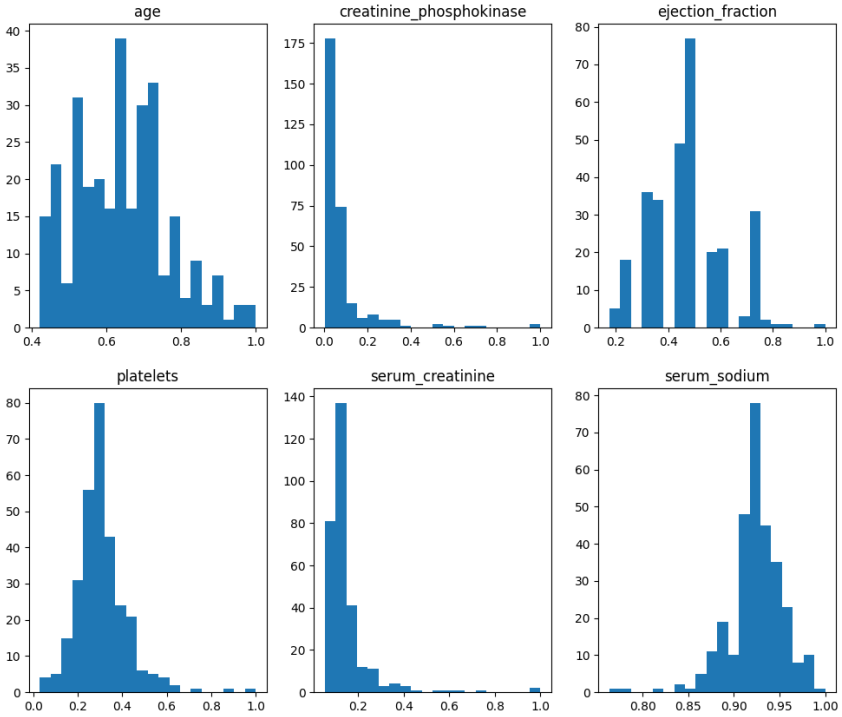

Рис. 4: Гистограмма признаков после MaxAbsScaler

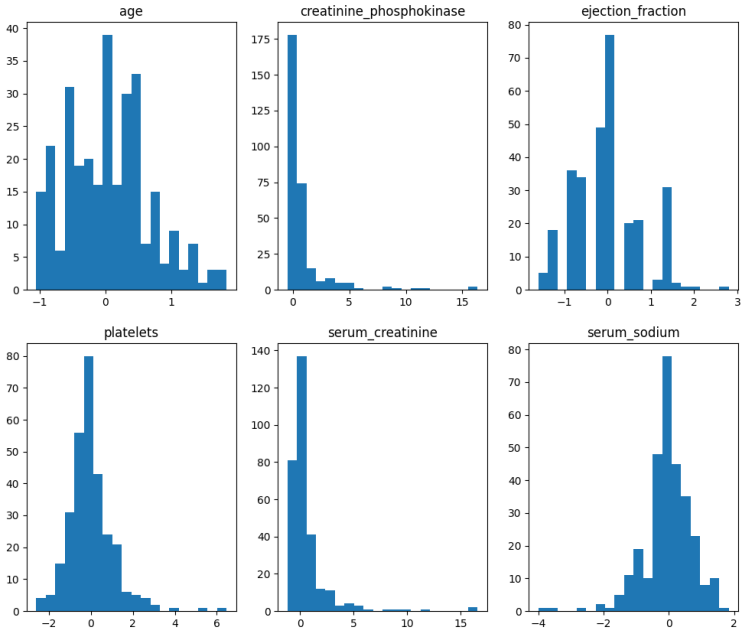

Рис. 5: Гистограмма признаков после RobustScaler

MaxAbsScaler приводит данные таким образом, что максимальное по модулю значение равно 1.

RobustScaler вычитает медиану и масшабирует данные в соответствии с межквартильным размахом.

3.5. Был написан вызов, который приводит данные к диапазону (-5, 15)

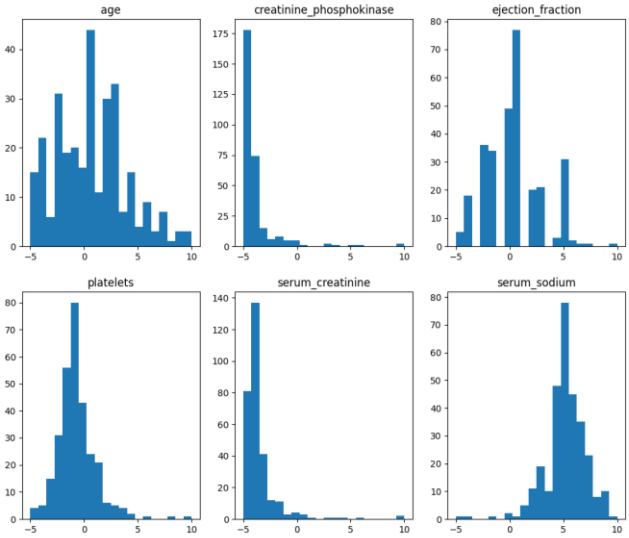

Рис. 6: Гистограмма признаков после MinMaxScaler((-5,10))

4. Нелинейные преобразования

4.1. При помощи QuantileTransformer данные были приведены к равномерному и нормальному распределениям.

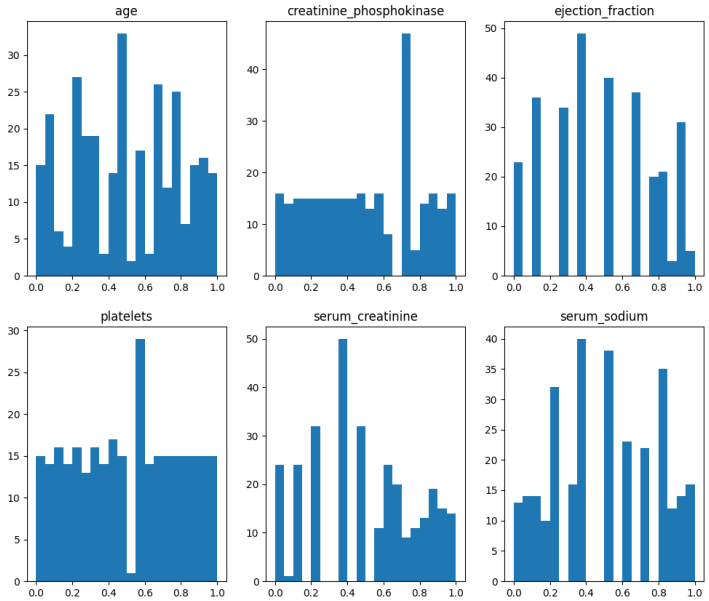

Рис. 7: Гистограмма равномерного распределения признаков после QuantileTransformer

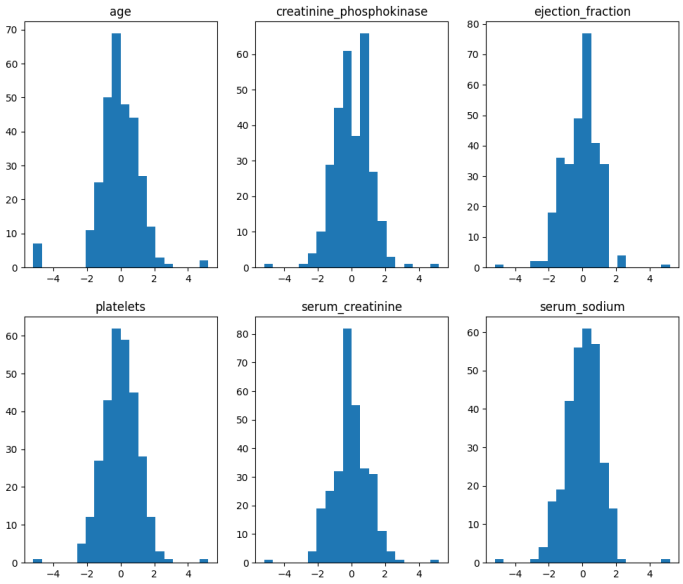

Рис. 8: Гистограмма нормального распределения признаков после QuantileTransformer

4.2. n_quantiles – параметр, указывающий количество квантилей, используемых для дискретизации функции распределения, чем больше количество квантилей, тем ближе полученная гистаграмма к требуемому распределению.

4.3. При помощи PowerTransformer данные были приведены к нормальному распределениям.

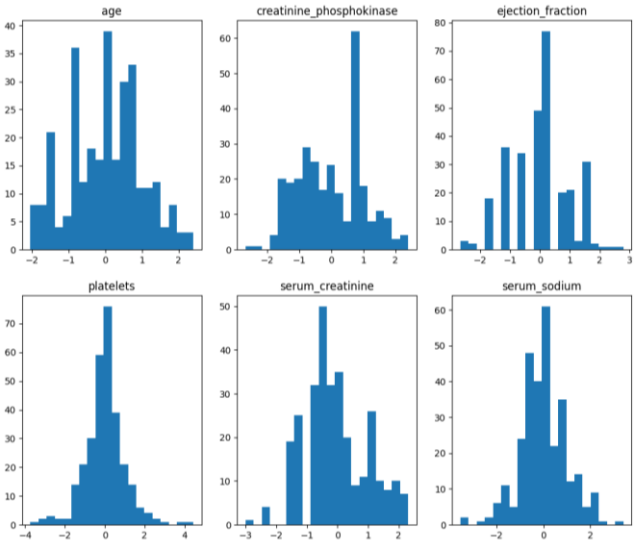

Рис. 9: Гистограмма нормального распределения признаков после PowerTransformer

5. Дискретизация признаков

5.1. Проведена дискретизация признаков, используя KBinsDiscretizer, на следующее количество диапазонов:

- age – 3
- creatinine_phosphokinase - 4
- ejection_fraction - 3
- platelets - 10
- serum_creatinine - 2
- serum_sodium – 4

5.2. Была построена гистограмма дискретизированных данных.

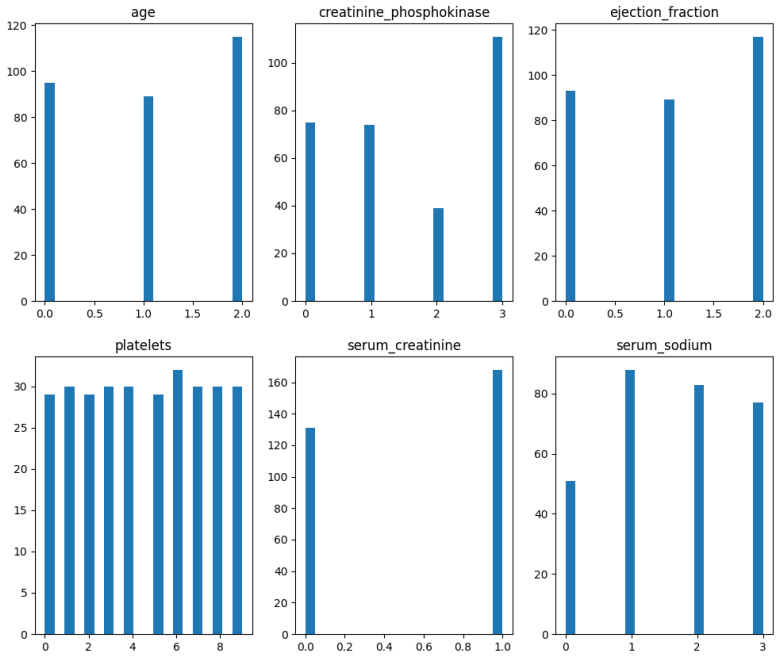

Рис. 10: Гистограмма признаков дискретизированных KBinsDiscretizer.

6. Выводы.

В ходе выполнения лабораторной работы изучены методы предобработки данных с помощью методов библиотеки scikit learn. При изучении стандартизации данных было выяснено, что при настройке на неполных данных происходит снижение качества результирующего набора данных, а приведение к диапазону не изменяет форму распределения. С помощью нелинейных преобразований преобразована изначальная форма распределения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

lab_work_1.py:

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

# Part1

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 наблю-
дений и 6 признаков

n_bins = 20
fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
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.savefig("fig1.png")

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

# Part2

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

fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
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.savefig("fig2.png")

scaler300 = preprocessing.StandardScaler().fit(data)
data_scaled300 = scaler300.transform(data)
```

```
print("Orig")
print("\n\n".join("%2.3e\n" % x + "%2.3e" % y for x, y in zip(numpy.mean(data,
0), numpy.std(data, 0))))
print("Std150")
print("\n\n".join("%2.3e\n" % x + "%2.3e" % y for x, y in
zip(numpy.mean(data_scaled, 0), numpy.std(data_scaled, 0))))
print("Std300")
print("\n\n".join(
    "%2.3e\n" % x + "%2.3e" % y for x, y in zip(numpy.mean(data_scaled300, 0),
numpy.std(data_scaled300, 0))))

# Part3

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

fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
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.savefig("fig3.png")

print("\t".join([str(x) for x in min_max_scaler.data_min_]))
print("\t".join([str(x) for x in min_max_scaler.data_max_]))

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

fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
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.savefig("fig4.png")

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

fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
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.savefig("fig5.png")

data_min_max_scaled_1 = preprocessing.MinMaxScaler((-5, 10)).fit_transform(data)

fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
axs[0, 0].hist(data_min_max_scaled_1[:, 0], bins=n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_min_max_scaled_1[:, 1], bins=n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_min_max_scaled_1[:, 2], bins=n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_min_max_scaled_1[:, 3], bins=n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_min_max_scaled_1[:, 4], bins=n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_min_max_scaled_1[:, 5], bins=n_bins)
axs[1, 2].set_title('serum_sodium')
plt.savefig("fig6.png")

# Part4

quantile_transformer = preprocessing.QuantileTransformer(n_quantiles=100,
                                                         ran-
dom_state=0).fit(data)
data_quantile_scaled = quantile_transformer.transform(data)

fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
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.savefig("fig7.png")

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

fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
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.savefig("fig8.png")

data_power_transformer_quantile_scaled = preprocessing.PowerTrans-
former().fit_transform(data)

fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
axs[0, 0].hist(data_power_transformer_quantile_scaled[:, 0], bins=n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(data_power_transformer_quantile_scaled[:, 1], bins=n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(data_power_transformer_quantile_scaled[:, 2], bins=n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(data_power_transformer_quantile_scaled[:, 3], bins=n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(data_power_transformer_quantile_scaled[:, 4], bins=n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(data_power_transformer_quantile_scaled[:, 5], bins=n_bins)
axs[1, 2].set_title('serum_sodium')
plt.savefig("fig9.png")

# Part 5

discretizer = preprocessing.KBinsDiscretizer(n_bins=[3, 4, 3, 10, 2, 4],
                                             encode='ordinal')
discretized_data = discretizer.fit_transform(data)

fig, axs = plt.subplots(2, 3)
fig.set_size_inches(12, 10)
axs[0, 0].hist(discretized_data[:, 0], bins=n_bins)
axs[0, 0].set_title('age')
axs[0, 1].hist(discretized_data[:, 1], bins=n_bins)
axs[0, 1].set_title('creatinine_phosphokinase')
axs[0, 2].hist(discretized_data[:, 2], bins=n_bins)
axs[0, 2].set_title('ejection_fraction')
axs[1, 0].hist(discretized_data[:, 3], bins=n_bins)
axs[1, 0].set_title('platelets')
axs[1, 1].hist(discretized_data[:, 4], bins=n_bins)
axs[1, 1].set_title('serum_creatinine')
axs[1, 2].hist(discretized_data[:, 5], bins=n_bins)
axs[1, 2].set_title('serum_sodium')
plt.savefig("fig10.png")

print('bin_edges_')
print(discretizer.bin_edges_)
```