Лабораторная работа №1 «Стандартизация данных»	Группа	ИВТ-363
	Студент	Сафошкин О.В.
	Дата выполнения	14.09.2023
	Преподаватель	Хайров А.В.
	Оценка	
	Подпись преподавателя	

Цель: познакомиться с методами стандартизации данных из библиотеки Scikit Learn.

Задачи:

1. Стандартизировать колонку из своего датасета используя sklearn предварительно преобразовав его в массив.
2. Стандартизировать все данные из датасета по строкам и по столбцам.
3. Привести данные к диапазону (0,3) используя MinMaxScaler.
4. Привести данные к диапазону (-1,1) используя MaxAbsScaler.
5. Привести данные к диапазону (25, 75) используя RobustScaler.
6. Привести данные к равномерному распределению используя QuantileTransformer.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
```

In [2]:
```
df = pd.read_csv('lab1.csv',encoding='cp1251')
array = np.array(df['age'])
normal_arr = preprocessing.normalize([array])
print("средний возраста до нормализации: ",df['age'].mean())
print("после: ",normal_arr.mean())
```

```
средний возраста до нормализации:  38.75757575757576
после:  0.06690918637379796
```

In [3]:
```
normal = preprocessing.normalize(df, axis=0)
print("ds до нормализации всех столбцов")
print(df.describe())
print("ds после нормализации")
print(pd.DataFrame(normal).describe())
```

```
ds до нормализации всех столбцов
               id         age      income  spending_rating
count  198.000000  198.000000  198.000000       198.000000
mean   100.424242   38.757576   60.893939        49.813131
std     57.487512   13.909126   25.576279        25.644988
min      1.000000   18.000000   15.000000         1.000000
25%     51.250000   28.250000   43.000000        34.250000
50%    100.500000   36.000000   62.000000        50.000000
75%    149.750000   48.750000   78.000000        72.000000
max    200.000000   70.000000  137.000000        99.000000
ds после нормализации
                0           1           2           3
count  198.000000  198.000000  198.000000  198.000000
mean     0.061715    0.066909    0.065547    0.063219
std      0.035328    0.024012    0.027531    0.032546
min      0.000615    0.031074    0.016146    0.001269
25%      0.031495    0.048769    0.046286    0.043467
50%      0.061761    0.062149    0.066738    0.063456
75%      0.092027    0.084160    0.083960    0.091376
max      0.122908    0.120845    0.147468    0.125642
```

In [4]:
```
scaler = preprocessing.MinMaxScaler()
d = scaler.fit_transform(df)
scal_df = pd.DataFrame(d, columns=df.columns)
scal_df.describe()
```

Out[4]:

	id	age	income	spending_rating
count	198.000000	198.000000	198.000000	198.000000
mean	0.499619	0.399184	0.376180	0.498093
std	0.288882	0.267483	0.209642	0.261684
min	0.000000	0.000000	0.000000	0.000000
25%	0.252513	0.197115	0.229508	0.339286
50%	0.500000	0.346154	0.385246	0.500000
75%	0.747487	0.591346	0.516393	0.724490
max	1.000000	1.000000	1.000000	1.000000

1. Стандартизировать колонку из своего датасета используя sklearn предварительно преобразовав его в массив.

In [5]:
```
arr_income = np.array(df['income'])
arr_income_norm = preprocessing.normalize([arr_income])
print("средний заработок до нормализации:",df['income'].mean())
print("после: ",arr_income_norm.mean())
```

```
средний заработок до нормализации: 60.89393939393939
после:  0.06554692435161912
```

2. Стандартизировать все данные из датасета по строкам и по столбцам.

In [6]:
```
normal_all = preprocessing.normalize(df)
print(df.describe())
print(pd.DataFrame(normal_all).describe())
```

```
               id         age      income  spending_rating
count  198.000000  198.000000  198.000000       198.000000
mean   100.424242   38.757576   60.893939        49.813131
std     57.487512   13.909126   25.576279        25.644988
min      1.000000   18.000000   15.000000         1.000000
25%     51.250000   28.250000   43.000000        34.250000
50%    100.500000   36.000000   62.000000        50.000000
75%    149.750000   48.750000   78.000000        72.000000
max    200.000000   70.000000  137.000000        99.000000
                0           1           2           3
count  198.000000  198.000000  198.000000  198.000000
mean     0.649367    0.321654    0.427692    0.393518
std      0.220550    0.181847    0.072853    0.225304
min      0.021780    0.103783    0.173013    0.005545
25%      0.547257    0.179022    0.409160    0.270152
50%      0.734912    0.268331    0.437439    0.390355
75%      0.809974    0.421338    0.459415    0.475030
max      0.890352    0.949112    0.867091    0.952546
```

3. Привести данные к диапазону (0,3) используя MinMaxScaler.

In [7]:
```
scaler2 = preprocessing.MinMaxScaler(feature_range=(0,3))
mima= scaler2.fit_transform(df)
scal_df = pd.DataFrame(mima, columns=df.columns)
scal_df.describe()
```

Out[7]:

	id	age	income	spending_rating
count	198.000000	198.000000	198.000000	198.000000
mean	1.498858	1.197552	1.128539	1.494280
std	0.866646	0.802450	0.628925	0.785051
min	0.000000	0.000000	0.000000	0.000000
25%	0.757538	0.591346	0.688525	1.017857
50%	1.500000	1.038462	1.155738	1.500000
75%	2.242462	1.774038	1.549180	2.173469
max	3.000000	3.000000	3.000000	3.000000

4. Привести данные к диапазону (-1,1) используя MaxAbsScaler.

- MaxAbsScaler is similar to MinMaxScaler except that the values are mapped across several ranges depending on whether negative OR positive values are present. If only positive values are present, the range is [0, 1]. Поэтому скелиться к [0,1]

In [8]:
```
abs_scaler = preprocessing.MaxAbsScaler()
m = abs_scaler.fit_transform(df)
abs_scal_df = pd.DataFrame(m, columns=df.columns)
abs_scal_df.describe()
```

Out[8]:

	id	age	income	spending_rating
count	198.000000	198.000000	198.000000	198.000000
mean	0.502121	0.553680	0.444481	0.503163
std	0.287438	0.198702	0.186688	0.259040
min	0.005000	0.257143	0.109489	0.010101
25%	0.256250	0.403571	0.313869	0.345960
50%	0.502500	0.514286	0.452555	0.505051
75%	0.748750	0.696429	0.569343	0.727273
max	1.000000	1.000000	1.000000	1.000000

5. Привести данные к диапазону (25, 75) используя RobustScaler.

- IQR определяется как разница между 75-м и 25-м перцентилями данных. Это означает, что IQR содержит в себе центральные 50% данных. данные масштабируются путем деления на IQR каждого признака. это делает данные более устойчивыми к наличию выбросов, предоставляя масштабированные данные, которые лучше соответствуют реальным свойствам данных в случае их асимметрии или наличия аномальных значений.

In [9]:
```
rob_scaler = preprocessing.RobustScaler(with_scaling=True)
r = rob_scaler.fit_transform(df)
rob_scal_df = pd.DataFrame(r, columns=df.columns)
rob_scal_df.describe()
```

Out[9]:

	id	age	income	spending_rating
count	198.000000	198.000000	198.000000	198.000000
mean	-0.000769	0.134516	-0.031602	-0.004950
std	0.583630	0.678494	0.730751	0.679337
min	-1.010152	-0.878049	-1.342857	-1.298013
25%	-0.500000	-0.378049	-0.542857	-0.417219
50%	0.000000	0.000000	0.000000	0.000000
75%	0.500000	0.621951	0.457143	0.582781
max	1.010152	1.658537	2.142857	1.298013

6. Привести данные к равномерному распределению используя QuantileTransformer.

In [10]:
```
un = preprocessing.QuantileTransformer()
u = un.fit_transform(df)
rob_scal_df = pd.DataFrame(u, columns=df.columns)
rob_scal_df.describe()
```

```
D:\.dev\Python\Lib\site-packages\sklearn\preprocessing\_data.py:2663: UserWarning: n
_quantiles (1000) is greater than the total number of samples (198). n_quantiles is
set to n_samples.
  warnings.warn(
```

Out[10]:

	id	age	income	spending_rating
count	198.000000	198.000000	198.000000	198.000000
mean	0.500000	0.500743	0.501077	0.500923
std	0.290872	0.291216	0.290942	0.290748
min	0.000000	0.000000	0.000000	0.000000
25%	0.250000	0.248731	0.258883	0.252538
50%	0.500000	0.505076	0.505076	0.510152
75%	0.750000	0.760787	0.776650	0.751269
max	1.000000	1.000000	1.000000	1.000000