Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ___«Информатика и системы управления» _____

КАФЕДРА _____«Системы обработки информации и управления» _____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

_____ *Типовое исследование* _____

Студент ___ИУ5-61Б__ _____ **_Павловская А.А.__**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ **__ Гапанюк Ю.Е.___**
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
(Индекс)
_____ _____
(И.О.Фамилия)
« _____ » _____ 2021 г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине __ «Технологии машинного обучения» _____

Студент группы ___ИУ5-61Б__

_____Павловская_Анастасия_Андреевна_____
(Фамилия, имя, отчество)

Тема курсовой работы _ Типовое исследование_____
_____
_____

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
___ учебная _____
Источник тематики (кафедра, предприятие, НИР) __кафедра_____

График выполнения работы: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Задание ____ решение задачи машинного обучения на основе материалов дисциплины. _____
_____
_____
_____
_____

Оформление курсовой работы:

Расчетно-пояснительная записка на _38_ листах формата А4.

_____
_____
_____
_____

Дата выдачи задания « ___ » _____ 2021 г.

Руководитель курсовой работы _____ **___Гапанюк Ю.Е.___**
(Подпись, дата) (И.О.Фамилия)

Студент _____ **___Павловская А.А.__**
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

Введение

Курсовой проект – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного обучения. Результатом курсового проекта является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

Курсовой проект опирается на знания, умения и владения, полученные студентом в рамках лекций и лабораторных работ по дисциплине.

В рамках курсового проекта возможно проведение типового или нетипового исследования.

- Типовое исследование - решение задачи машинного обучения на основе материалов дисциплины. Выполняется студентом единолично.
- Нетиповое исследование - решение нестандартной задачи. Тема должна быть согласована с преподавателем. Как правило, такая работа выполняется группой студентов.

Основная часть

1. Постановка задачи

Схема типового исследования, проводимого студентом в рамках курсовой работы, содержит выполнение следующих шагов:

1) Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.
2) Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.
3) Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
4) Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения. В зависимости от набора данных, порядок выполнения пунктов 2, 3, 4 может быть изменен.
5) Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.
6) Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.
7) Формирование обучающей и тестовой выборок на основе исходного набора данных.
8) Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.
9) Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.

10) Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.
11) Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется построение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

Разработка макета веб-приложения, предназначенного для анализа данных.

- Вариант 2. Макет должен быть реализован для нескольких моделей машинного обучения. Макет должен позволять:
    - выбирать модели для обучения,
    - производить обучение,
    - осуществлять просмотр результатов обучения, в том числе в виде графиков.

2. Последовательность действий по решению поставленной задачи

Последовательность действий по решению поставленной задачи представлена в файле KR_TMO.ipynb.

ИУ5-61Б Павловская А.А.

Курсовая работа по дисциплине "Технологии машинного обучения"

Поиск и выбор набора данных для построения моделей машинного обучения

В качестве набора данных мы будем использовать набор данных **Life Expectancy (WHO) -** https://www.kaggle.com/kumarajarshi/life-expectancy-who

Набор данных предназначен для статистического анализа факторов, влияющих на ожидаемую продолжительность жизни.

Датасет состоит из одного файла:

- **Life Expectancy Data.csv**

Файл содержит следующие колонки:

- **Country -** страна
- **Year -** год
- **Status -** статус страны "развитая" или "развивающаяся"
- **Life_expectancy -** ожидаемая продолжительность жизни (возраст)
- **Adult_Mortality -** показатели смертности взрослого населения обоих полов (вероятность смерти в возрасте от **15** до **60** лет на **1000** населения)
- **infant_deaths -** число младенческих смертей на **1000** населения
- **Alcohol -** алкоголь, зарегистрированное потребление на душу населения **(15+)** (в литрах чистого алкоголя)
- **percentage_expenditure -** расходы на здравоохранение в процентах от валового внутреннего продукта на душу населения**(%)**
- **Hepatitis_B -** охват иммунизацией против гепатита B **(HepB)** среди детей в возрасте **1** года **(%)**
- **Measles -** корь **-** количество зарегистрированных случаев заболевания на **1000** человек населения
- **BMI -** средний индекс массы тела всего населения
- **under-five_deaths -** число смертей в возрасте до пяти лет на **1000** человек населения
- **Polio -** охват иммунизацией против полиомиелита **(Pol3)** среди детей в возрасте **1** года **(%)**
- **Total_expenditure -** общие государственные расходы на здравоохранение в процентах от общего объема государственных расходов **(%)**
- **Diphtheria -** охват иммунизацией против столбняка дифтерии и коклюша **(DTP3)** среди детей в возрасте **1** года **(%)**
- **HIV/AIDS -** смертность на **1 000** живорождений ВИЧ/СПИД **(0-4** года**)**
- **GDP -** валовой внутренний продукт на душу населения (в долларах США)
- **Population -** население страны
- **thinness_1-19_years -** распространенность худобы среди детей и подростков в возрасте от **10** до **19** лет **(%)**
- **thinness_5-9_years -** распространенность худобы среди детей в возрасте от **5** до **9** лет**(%)**
- **Income_composition_of_resources -** индекс человеческого развития с точки зрения структуры доходов от ресурсов (индекс в диапазоне от **0** до **1)**
- **Schooling -** количество лет обучения в школе(лет)

В данной работе будет решаться задача регрессии. В качестве целевого признака будет использован признак **"Life_expectancy".**

Импорт библиотек

In [137]:

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_err
or, median_absolute_error, r2_score
from sklearn.tree import DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import  RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import SGDRegressor
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Загрузка данных

Загрузим файлы датасета в помощью библиотеки **Pandas.**

In [138]:

```
data = pd.read_csv('data/Life Expectancy Data.csv')
```

In [139]:

```
# Удаление дубликатов записей, если они присутствуют
data.drop_duplicates()
```

Out[139]:

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatit
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	
...	...	...	...	...	...	...	...	...	
2933	Zimbabwe	2004	Developing	44.3	723.0	27	4.36	0.000000	
2934	Zimbabwe	2003	Developing	44.5	715.0	26	4.06	0.000000	
2935	Zimbabwe	2002	Developing	44.8	73.0	25	4.43	0.000000	
2936	Zimbabwe	2001	Developing	45.3	686.0	25	1.72	0.000000	
2937	Zimbabwe	2000	Developing	46.0	665.0	24	1.68	0.000000	

2938 rows × 22 columns

Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Кодирование категориальных признаков. Анализ и заполнение пропусков в данных.

Основные характеристики датасета

In [140]:

```
# Первые пять строк
data.head()
```

Out[140]:

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.

5 rows × 22 columns

In [141]:

```
# Список колонок
data.columns
```

Out[141]:

```
Index(['Country', 'Year', 'Status', 'Life_expectancy', 'Adult_Mortality',
       'infant_deaths', 'Alcohol', 'percentage_expenditure', 'Hepatitis_B',
       'Measles', 'BMI', 'under-five_deaths', 'Polio', 'Total_expenditure',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness_1-19_years',
       'thinness_5-9_years', 'Income_composition_of_resources', 'Schooling'],
      dtype='object')
```

In [142]:

```
# Список колонок с типами данных
data.dtypes
```

Out[142]:

```
Country                           object
Year                               int64
Status                            object
Life_expectancy                  float64
Adult_Mortality                  float64
infant_deaths                      int64
Alcohol                          float64
percentage_expenditure           float64
Hepatitis_B                      float64
Measles                            int64
BMI                              float64
under-five_deaths                  int64
Polio                            float64
Total_expenditure                float64
Diphtheria                       float64
HIV/AIDS                         float64
GDP                              float64
Population                       float64
thinness_1-19_years              float64
thinness_5-9_years               float64
Income_composition_of_resources  float64
Schooling                        float64
dtype: object
```

In [143]:

```
data.shape
```

Out[143]:

```
(2938, 22)
```

In [144]:

```
# Кодирование категориальных признаков целочисленными значениями
le = LabelEncoder()
le.fit(data.Status)
data.Status = le.transform(data.Status)
le.fit(data.Country)
data.Country = le.transform(data.Country)
```

In [145]:

```
data.head()
```

Out[145]:

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measl
0	0	2015	1	65.0	263.0	62	0.01	71.279624	65.0	11
1	0	2014	1	59.9	271.0	64	0.01	73.523582	62.0	4
2	0	2013	1	59.9	268.0	66	0.01	73.219243	64.0	4
3	0	2012	1	59.5	272.0	69	0.01	78.184215	67.0	27
4	0	2011	1	59.2	275.0	71	0.01	7.097109	68.0	30

5 rows × 22 columns

In [146]:

```
data.dtypes
```

Out[146]:

```
Country                          int32
Year                             int64
Status                           int32
Life_expectancy                float64
Adult_Mortality                float64
infant_deaths                    int64
Alcohol                        float64
percentage_expenditure         float64
Hepatitis_B                    float64
Measles                          int64
BMI                            float64
under-five_deaths                int64
Polio                          float64
Total_expenditure              float64
Diphtheria                     float64
HIV/AIDS                       float64
GDP                            float64
Population                     float64
thinness_1-19_years            float64
thinness_5-9_years             float64
Income_composition_of_resources    float64
Schooling                      float64
dtype: object
```

In [147]:

```
# Проверка наличия пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
```

```
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
Country - 0
Year - 0
Status - 0
Life_expectancy - 10
Adult_Mortality - 10
infant_deaths - 0
Alcohol - 194
percentage_expenditure - 0
Hepatitis_B - 553
Measles - 0
BMI - 34
under-five_deaths - 0
Polio - 19
Total_expenditure - 226
Diphtheria - 19
HIV/AIDS - 0
GDP - 448
Population - 652
thinness_1-19_years - 34
thinness_5-9_years - 34
Income_composition_of_resources - 167
Schooling - 163
```

In [148]:

```
# Выбор числовых колонок с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / data.shape[0]) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(co
l, dt, temp_null_count, temp_perc))
```

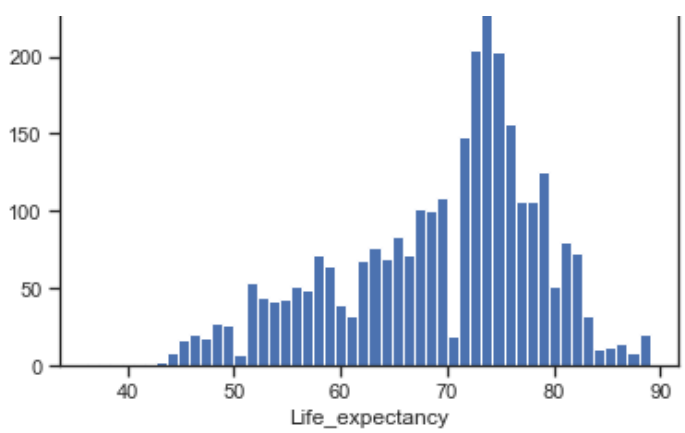
```
Колонка Life_expectancy. Тип данных float64. Количество пустых значений 10, 0.34%.
Колонка Adult_Mortality. Тип данных float64. Количество пустых значений 10, 0.34%.
Колонка Alcohol. Тип данных float64. Количество пустых значений 194, 6.6%.
Колонка Hepatitis_B. Тип данных float64. Количество пустых значений 553, 18.82%.
Колонка BMI. Тип данных float64. Количество пустых значений 34, 1.16%.
Колонка Polio. Тип данных float64. Количество пустых значений 19, 0.65%.
Колонка Total_expenditure. Тип данных float64. Количество пустых значений 226, 7.69%.
Колонка Diphtheria. Тип данных float64. Количество пустых значений 19, 0.65%.
Колонка GDP. Тип данных float64. Количество пустых значений 448, 15.25%.
Колонка Population. Тип данных float64. Количество пустых значений 652, 22.19%.
Колонка thinness_1-19_years. Тип данных float64. Количество пустых значений 34, 1.16%.
Колонка thinness_5-9_years. Тип данных float64. Количество пустых значений 34, 1.16%.
Колонка Income_composition_of_resources. Тип данных float64. Количество пустых значений 1
67, 5.68%.
Колонка Schooling. Тип данных float64. Количество пустых значений 163, 5.55%.
```

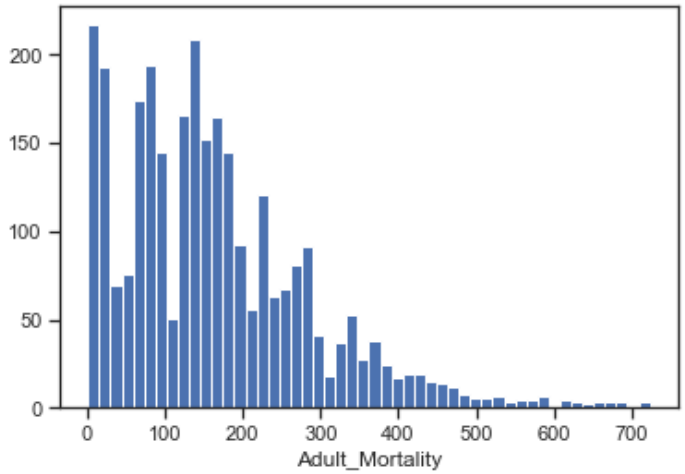
In [149]:

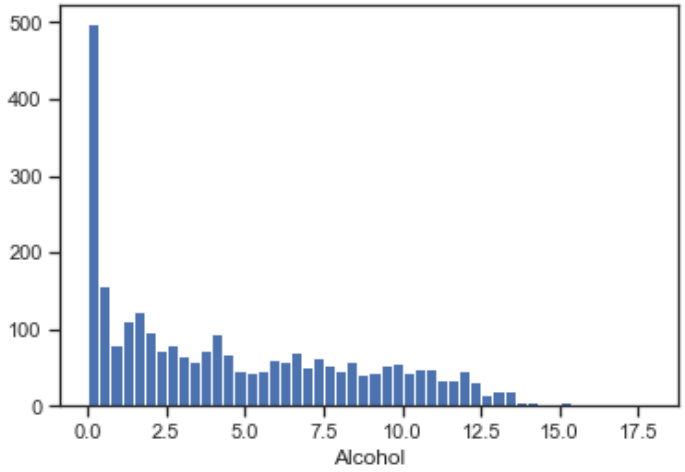
```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
```

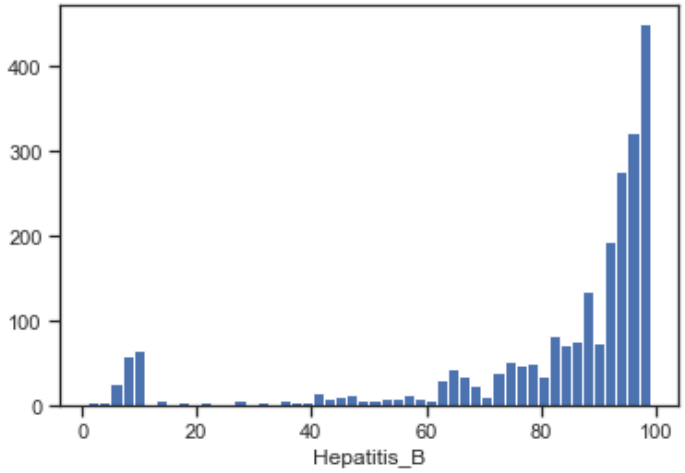
In [150]:

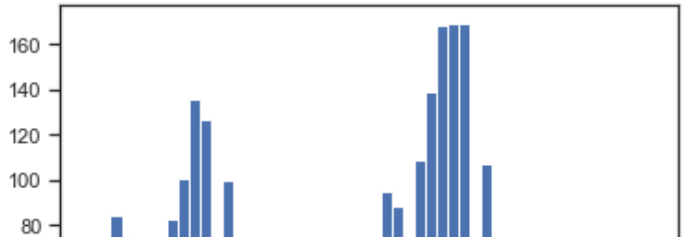
```
# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```

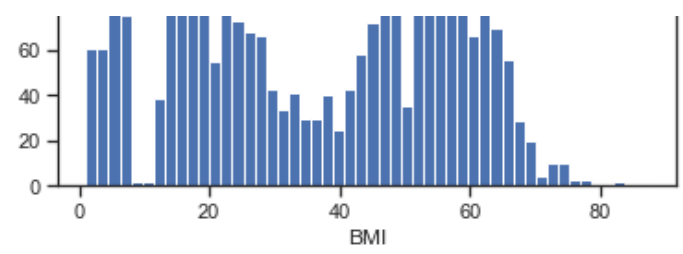

200
150
100
50
0
40
50
60
70
80
90
Life_expectancy

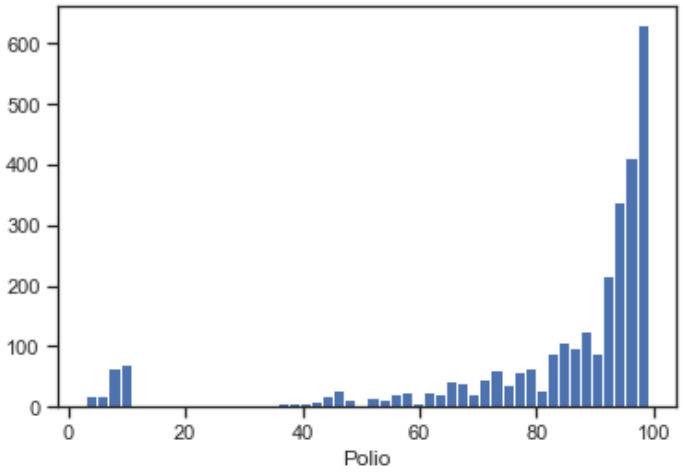

200
150
100
50
0
0
100
200
300
400
500
600
700
Adult_Mortality

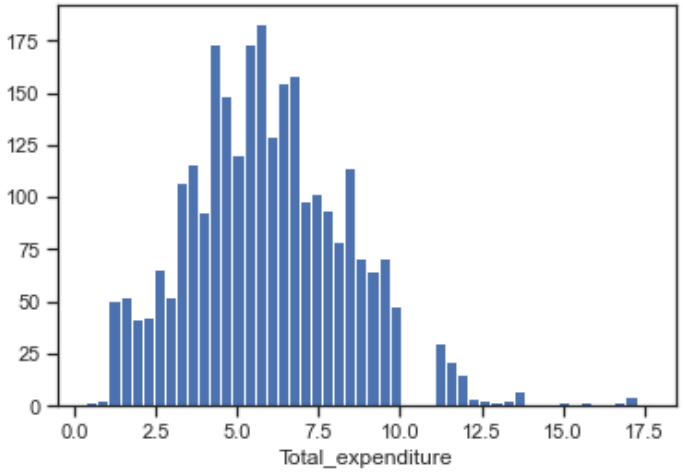

500
400
300
200
100
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Alcohol

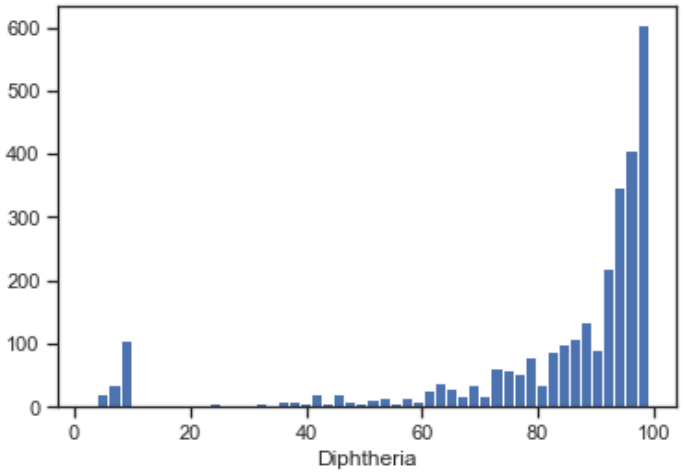

400
300
200
100
0
0
20
40
60
80
100
Hepatitis_B

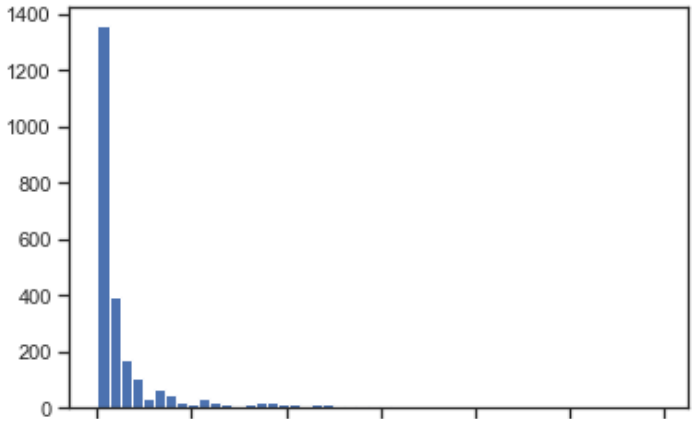

160
140
120
100
80


0
20
40
60
80
BMI

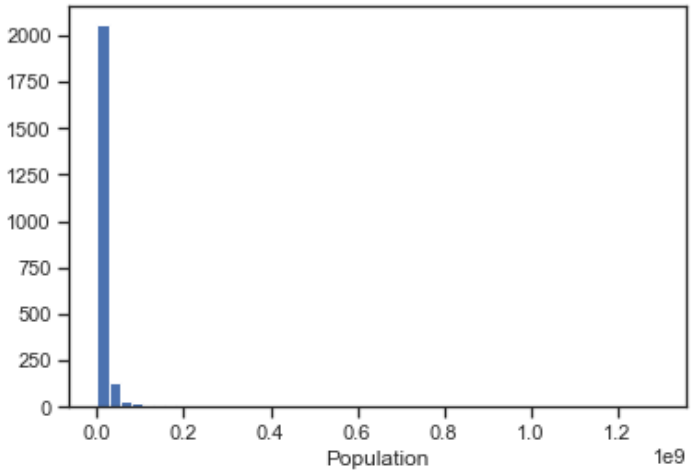

0
100
200
300
400
500
600
0
20
40
60
80
100
Polio

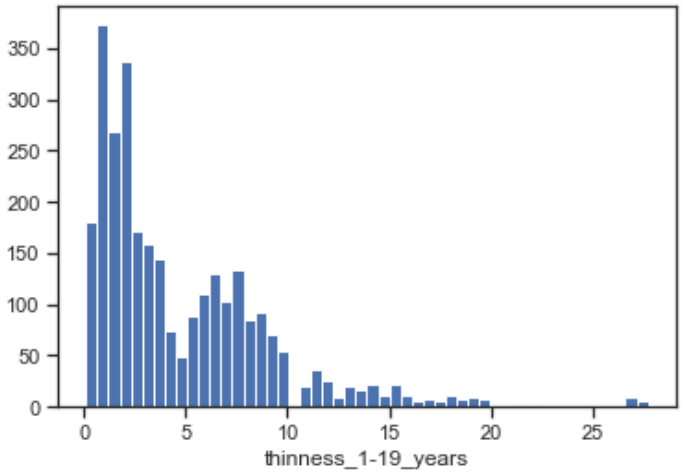

0
25
50
75
100
125
150
175
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Total_expenditure

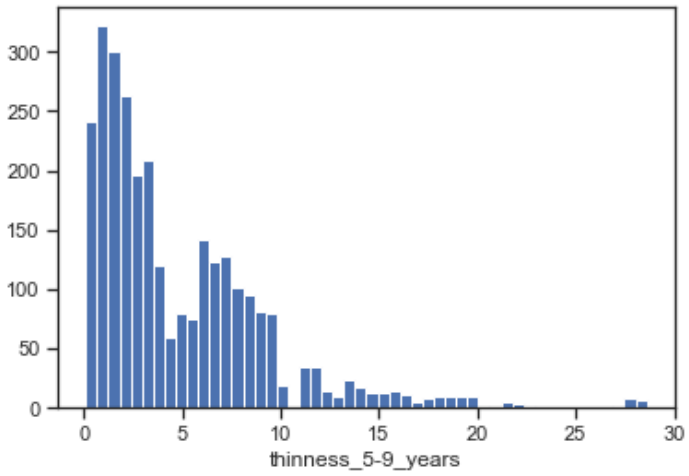

0
100
200
300
400
500
600
0
20
40
60
80
100
Diphtheria

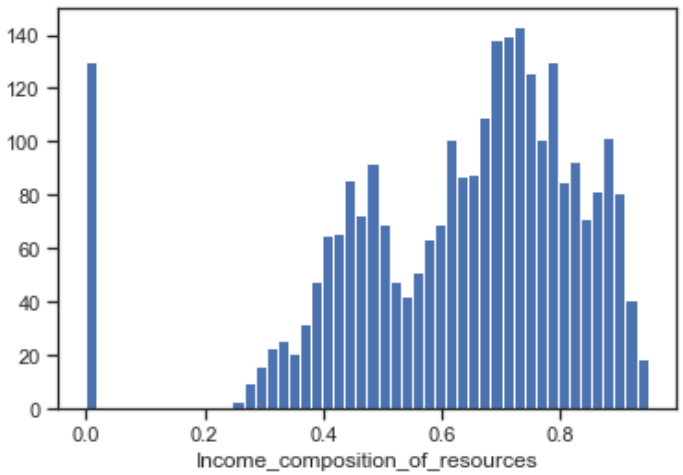

0
200
400
600
800
1000
1200
1400


0
20000
40000
60000
80000
100000
120000
GDP

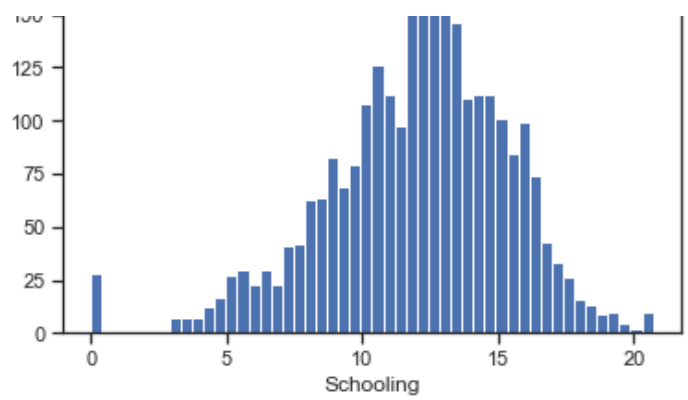

2000
1750
1500
1250
1000
750
500
250
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Population
1e9


350
300
250
200
150
100
50
0
0
5
10
15
20
25
thinness_1-19_years


300
250
200
150
100
50
0
0
5
10
15
20
25
30
thinness_5-9_years


140
120
100
80
60
40
20
0
0.0
0.2
0.4
0.6
0.8
Income_composition_of_resources


175
150



In [151]:

```
# Удаление колонок со слишком большим количеством пропусков
data.drop(['Hepatitis_B','GDP','Population',], axis = 1, inplace = True)
```

In [152]:

```
# Обработка пропусков
imp_num = SimpleImputer(strategy='median')
imp_num2 = SimpleImputer(strategy='most_frequent')
data[['Life_expectancy']] = imp_num2.fit_transform(data[['Life_expectancy']])
data[['Adult_Mortality']] = imp_num.fit_transform(data[['Adult_Mortality']])
data[['Alcohol']] = imp_num.fit_transform(data[['Alcohol']])
data[['BMI']] = imp_num.fit_transform(data[['BMI']])
data[['Polio']] = imp_num.fit_transform(data[['Polio']])
data[['Total_expenditure']] = imp_num2.fit_transform(data[['Total_expenditure']])
data[['Diphtheria']] = imp_num.fit_transform(data[['Diphtheria']])
data[['thinness_1-19_years']] = imp_num.fit_transform(data[['thinness_1-19_years']])
data[['thinness_5-9_years']] = imp_num.fit_transform(data[['thinness_5-9_years']])
data[['Income_composition_of_resources']] = imp_num.fit_transform(data[['Income_compositi
on_of_resources']])
data[['Schooling']] = imp_num2.fit_transform(data[['Schooling']])
```

In [153]:

```
# Проверка наличия пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
Country - 0
Year - 0
Status - 0
Life_expectancy - 0
Adult_Mortality - 0
infant_deaths - 0
Alcohol - 0
percentage_expenditure - 0
Measles - 0
BMI - 0
under-five_deaths - 0
Polio - 0
Total_expenditure - 0
Diphtheria - 0
HIV/AIDS - 0
thinness_1-19_years - 0
thinness_5-9_years - 0
Income_composition_of_resources - 0
Schooling - 0
```

Все пропуски в данных заполнены.

Выбор признаков, подходящих для построения моделей. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.

In [154]:

```
# Числовые колонки для масштабирования
scale_cols = ['Life_expectancy', 'Adult_Mortality', 'infant_deaths', 'percentage_expenditure', 'Measles',
              'BMI','under-five_deaths','Polio','Total_expenditure','Diphtheria','thinness_1-19_years',
             'thinness_5-9_years','Schooling']
```

In [155]:

```
# Масштабирование данных
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[scale_cols])
```

In [156]:

```
# Добавим масштабированные данные в набор данных
for i in range(len(scale_cols)):
    col = scale_cols[i]
    new_col_name = col + '_scaled'
    data[new_col_name] = sc1_data[:,i]
```

In [157]:

```
data.head()
```

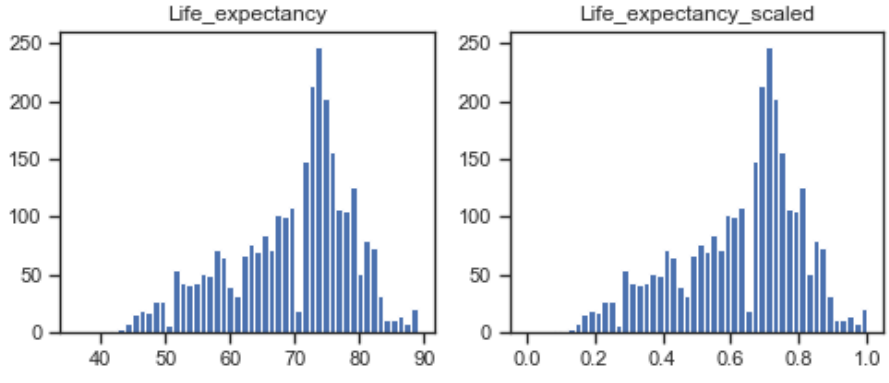
Out[157]:

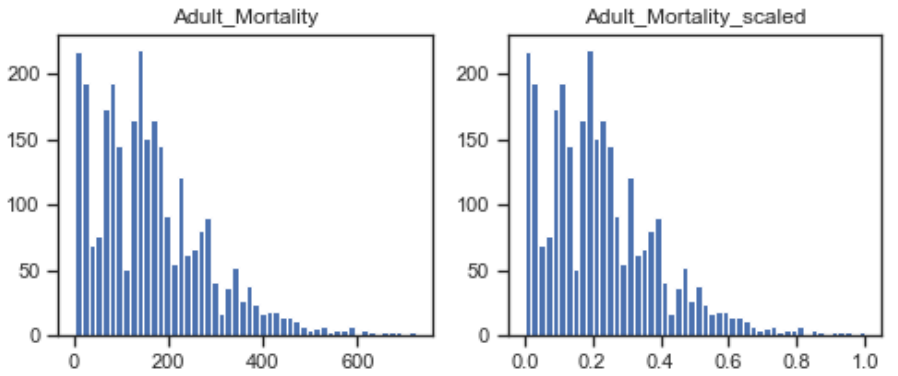
	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Measles	BMI	...
0	0	2015	1	65.0	263.0	62	0.01	71.279624	1154	19.1	...
1	0	2014	1	59.9	271.0	64	0.01	73.523582	492	18.6	...
2	0	2013	1	59.9	268.0	66	0.01	73.219243	430	18.1	...
3	0	2012	1	59.5	272.0	69	0.01	78.184215	2787	17.6	...
4	0	2011	1	59.2	275.0	71	0.01	7.097109	3013	17.2	...

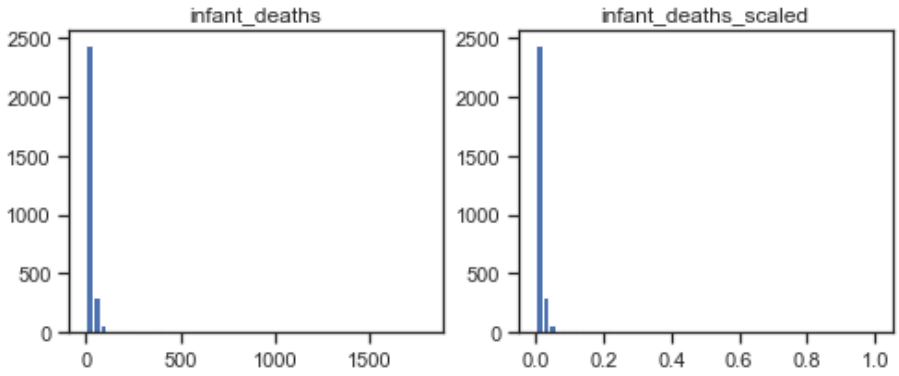
5 rows × 32 columns

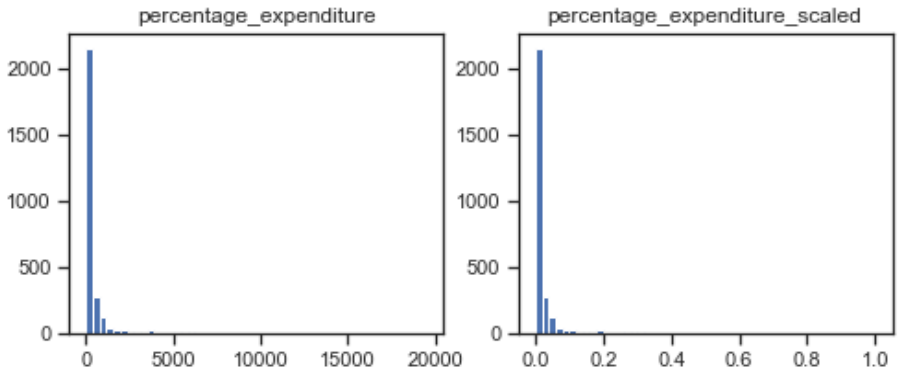
In [158]:

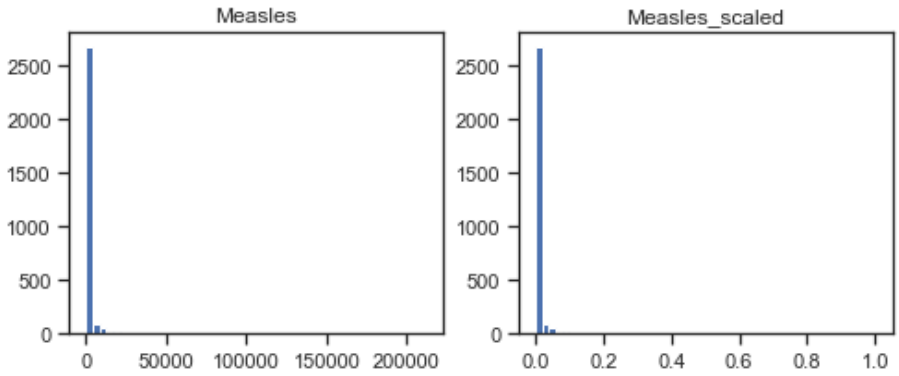
```
# Проверим, что масштабирование не повлияло на распределение данных
for col in scale_cols:
    col_scaled = col + '_scaled'

    fig, ax = plt.subplots(1, 2, figsize=(8,3))
    ax[0].hist(data[col], 50)
    ax[1].hist(data[col_scaled], 50)
    ax[0].title.set_text(col)
    ax[1].title.set_text(col_scaled)
    plt.show()
```

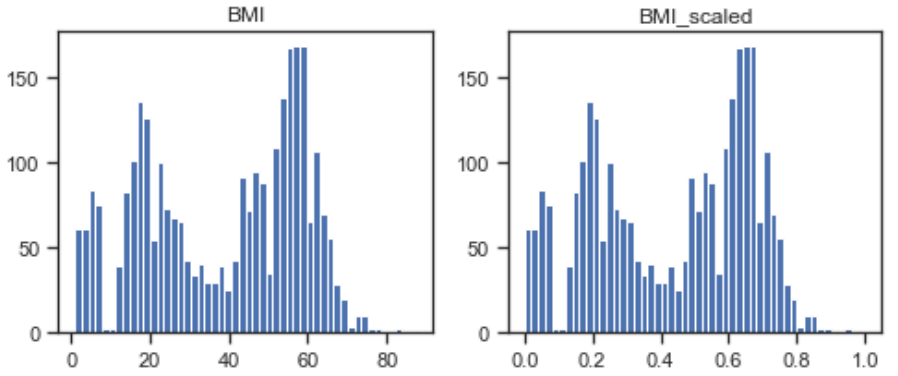


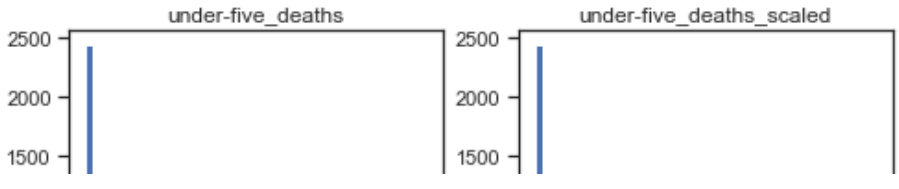

Adult_Mortality
Adult_Mortality_scaled

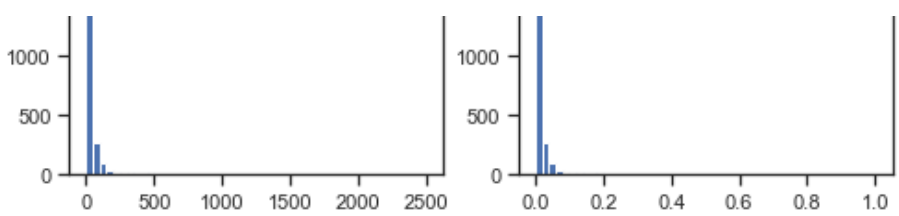

infant_deaths
infant_deaths_scaled

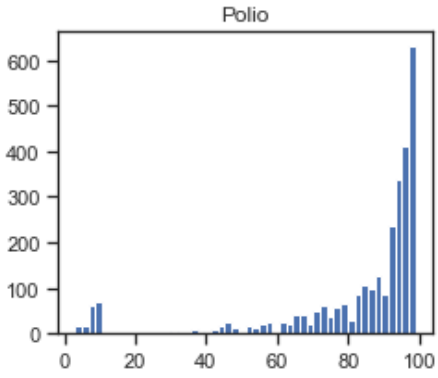

percentage_expenditure
percentage_expenditure_scaled

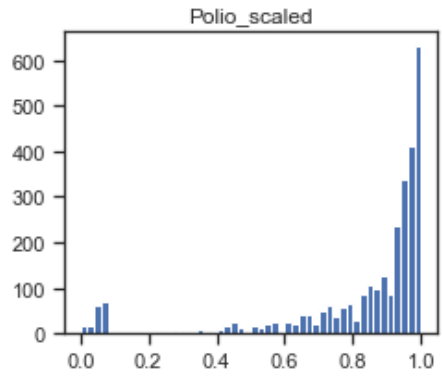

Measles
Measles_scaled

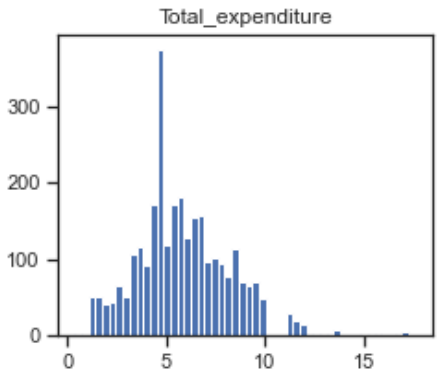

BMI
BMI_scaled

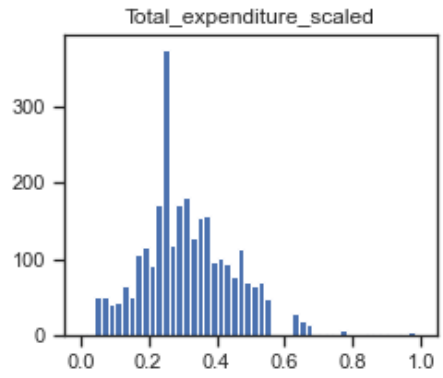

under-five_deaths
under-five_deaths_scaled

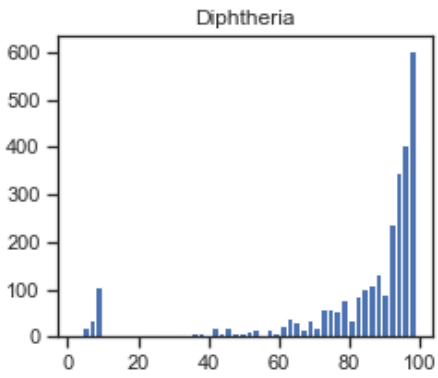


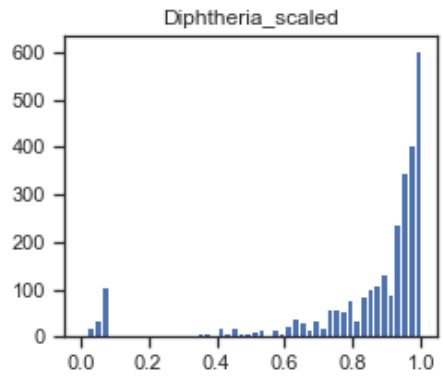

Polio

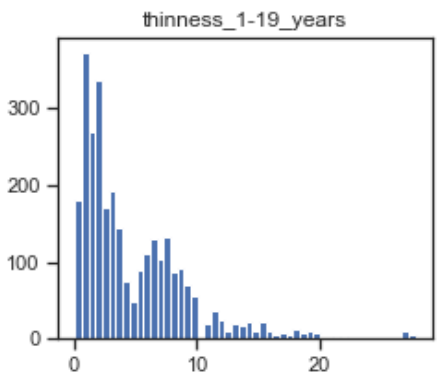

Polio_scaled

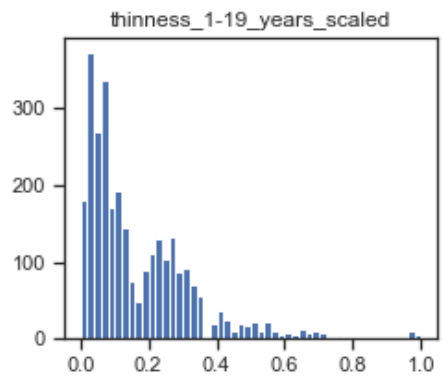

Total_expenditure

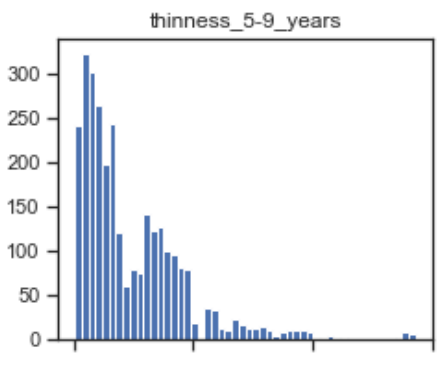

Total_expenditure_scaled

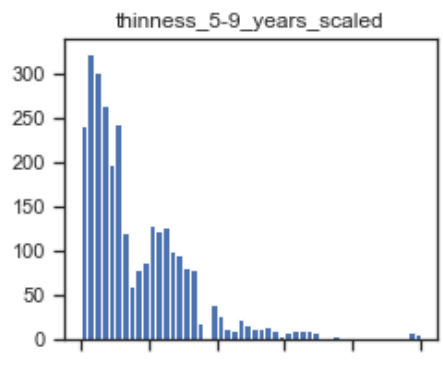

Diphtheria

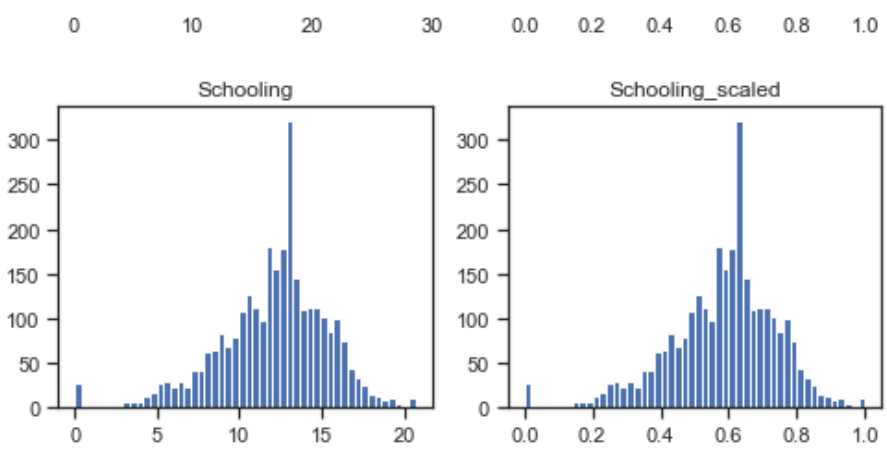

Diphtheria_scaled


thinness_1-19_years


thinness_1-19_years_scaled


thinness_5-9_years


thinness_5-9_years_scaled



Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения.

In [159]:

```
corr_cols_1 = scale_cols
corr_cols_1
```

Out[159]:

```
['Life_expectancy',
 'Adult_Mortality',
 'infant_deaths',
 'percentage_expenditure',
 'Measles',
 'BMI',
 'under-five_deaths',
 'Polio',
 'Total_expenditure',
 'Diphtheria',
 'thinness_1-19_years',
 'thinness_5-9_years',
 'Schooling']
```

In [160]:

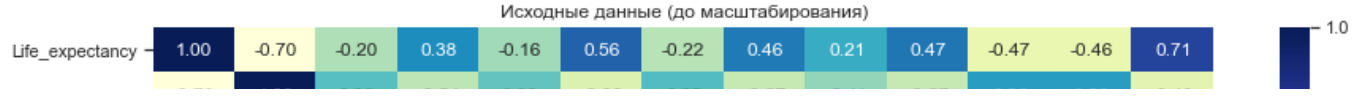
```
scale_cols_postfix = [x+'_scaled' for x in scale_cols]
corr_cols_2 = scale_cols_postfix
corr_cols_2
```

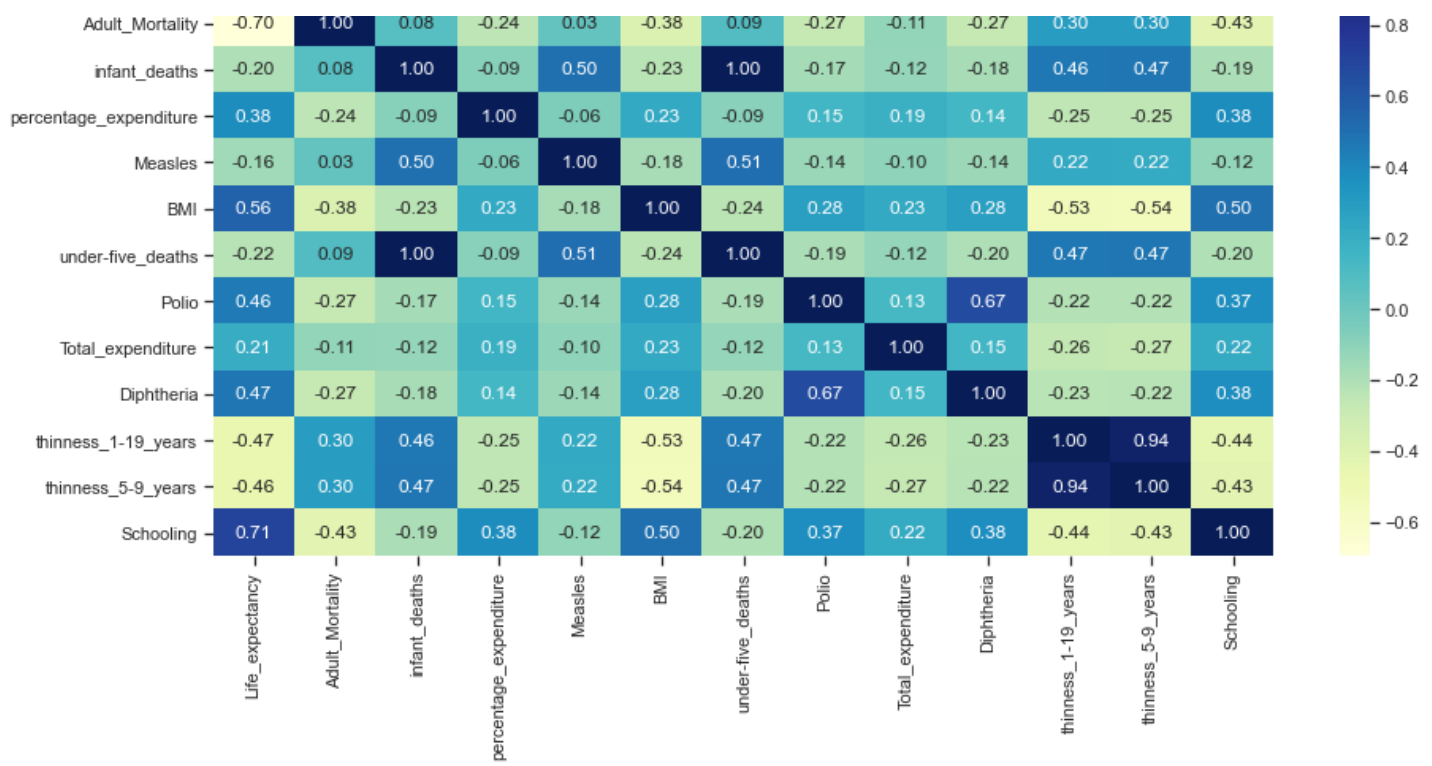
Out[160]:

```
['Life_expectancy_scaled',
 'Adult_Mortality_scaled',
 'infant_deaths_scaled',
 'percentage_expenditure_scaled',
 'Measles_scaled',
 'BMI_scaled',
 'under-five_deaths_scaled',
 'Polio_scaled',
 'Total_expenditure_scaled',
 'Diphtheria_scaled',
 'thinness_1-19_years_scaled',
 'thinness_5-9_years_scaled',
 'Schooling_scaled']
```

In [161]:

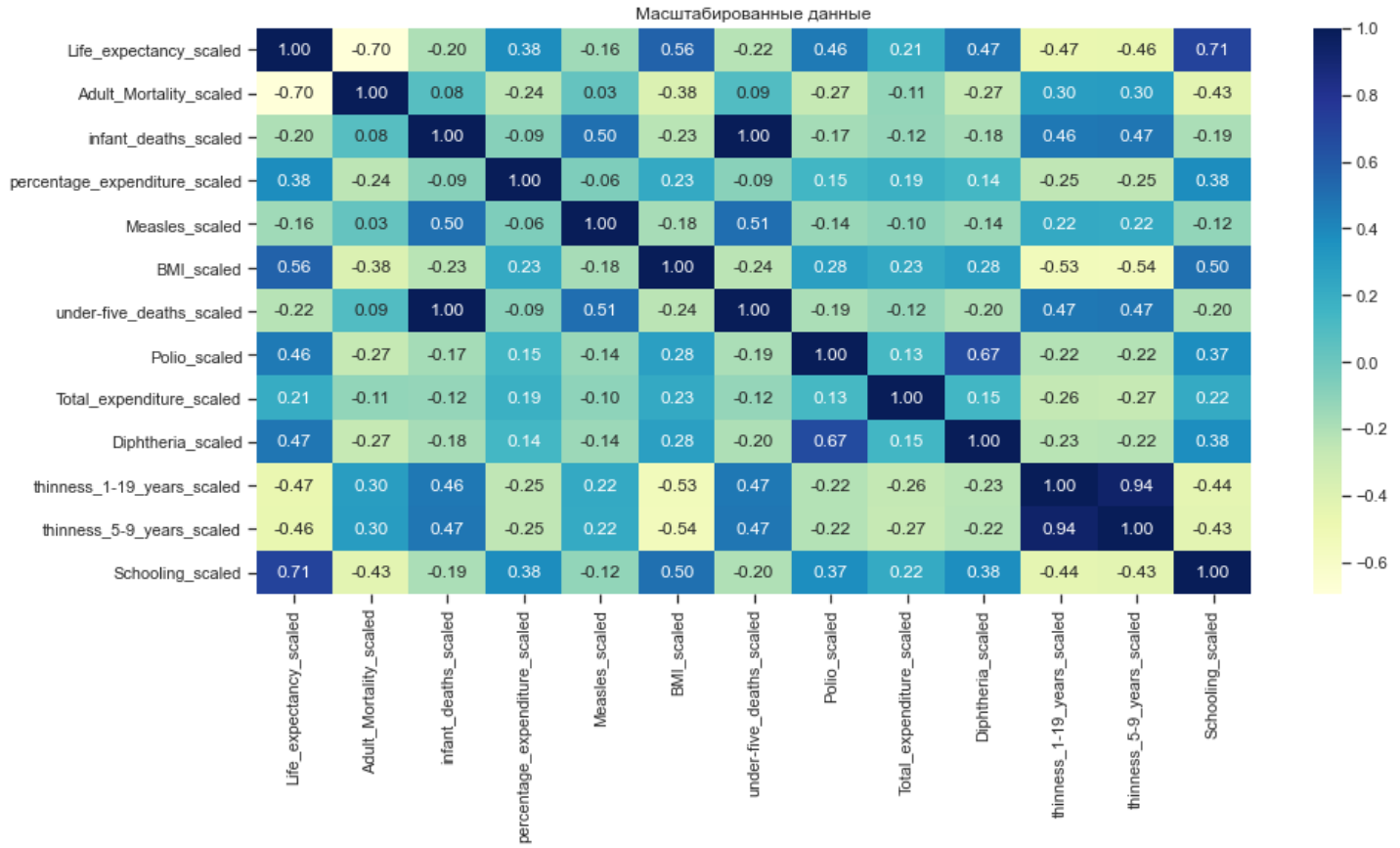
```
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(data[corr_cols_1].corr(),cmap='YlGnBu', annot=True, fmt='.2f')
ax.set_title('Исходные данные (до масштабирования)')
plt.show()
```





In [162]:

```
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(data[corr_cols_2].corr(),cmap='YlGnBu', annot=True, fmt='.2f')
ax.set_title('Масштабированные данные')
plt.show()
```



На основе корреляционной матрицы можно сделать следующие выводы:

- Корреляционные матрицы для исходных и масштабированных данных совпадают.
- Целевой признак регрессии **"Life_expectancy"** наиболее сильно коррелирует с **"Schooled" (0.71)** и **"BMI" (0.56).** Эти признаки обязательно следует оставить в модели регрессии.
- Пары признаков **"thinness_1-19_years"** и **"thinness_5-9_years", "under-five_deaths"** и **"infant_deaths"** имеют корреляцию, близкую по модулю к **1,** поэтому одновременно оба признака в одной паре не следуют включать в модели.
- Для построения моделей будем использовать признаки **"percentage_expenditure", "BMI"**

- Для построения моделей будем использовать признаки **"percentage_expenditure", "BMI", "Diphtheria","Polio","Total_expenditure", "Schooling"** по причине достаточно большой корреляции с целевым признаком.
- Большие по модулю значения коэффициентов корреляции свидетельствуют о значимой корреляции между исходными признаками и целевым признаком. На основании корреляционной матрицы можно сделать вывод о том, что данные позволяют построить модель машинного обучения.

Построение графиков для понимания структуры данных

In [163]:

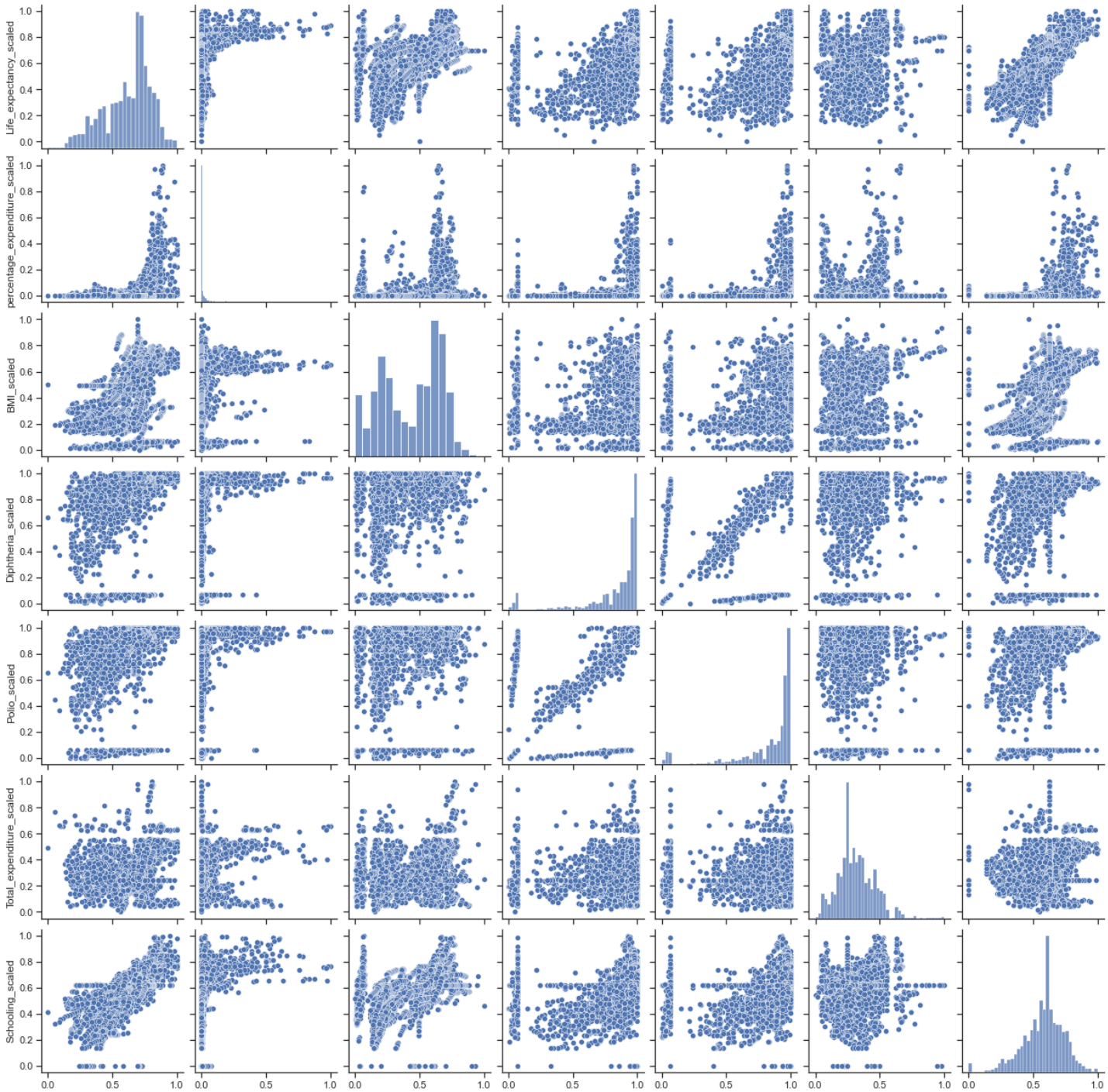
```
data_show = data[['Life_expectancy_scaled','percentage_expenditure_scaled','BMI_scaled','
Diphtheria_scaled',
              'Polio_scaled', 'Total_expenditure_scaled','Schooling_scaled']]
```

In [164]:

```
# Парные диаграммы
sns.pairplot(data_show)
```

Out[164]:

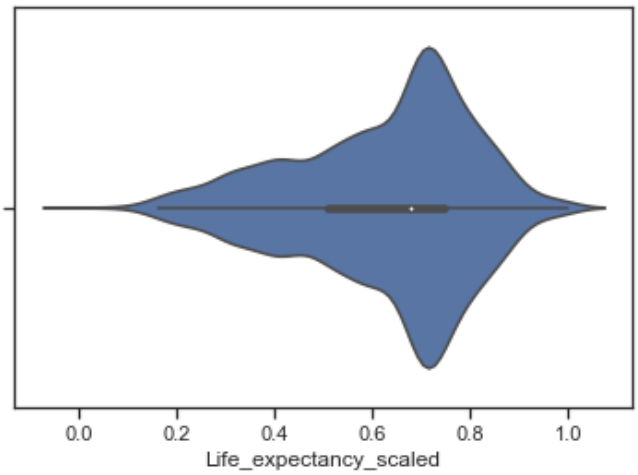
```
<seaborn.axisgrid.PairGrid at 0x16cf4c9c6a0>
```

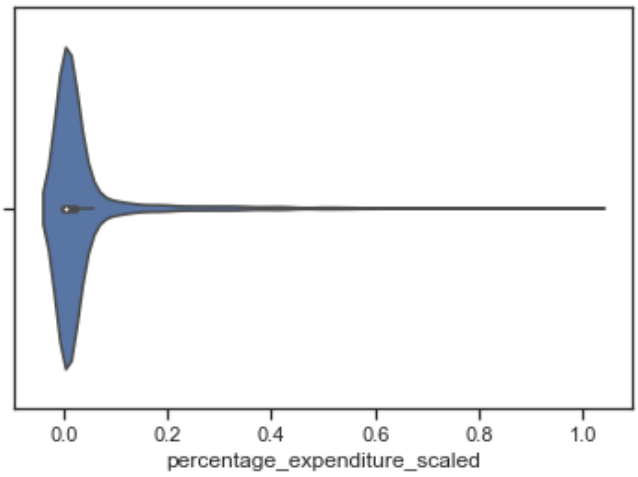


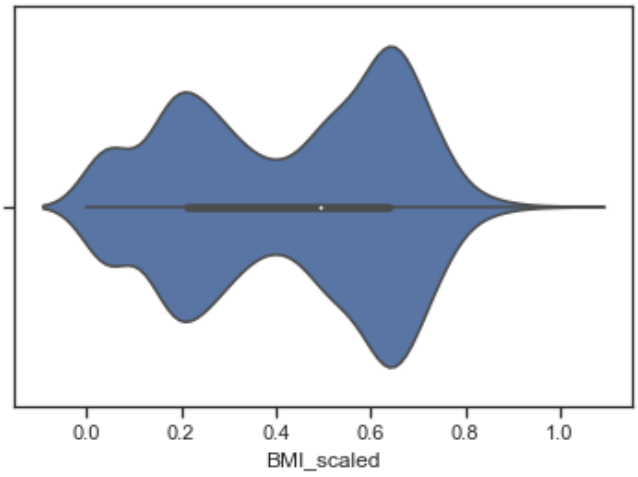
Life_expectancy_scaled percentage_expenditure_scaled BMI_scaled Diphtheria_scaled Polio_scaled Total_expenditure_scaled Schooling_scaled

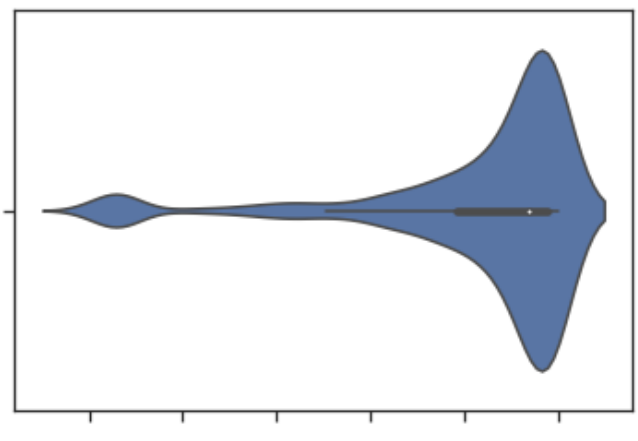
In [165]:

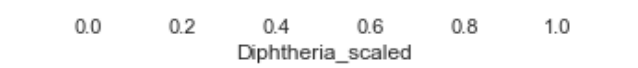
```
# Скрипичные диаграммы для числовых колонок
for col in data_show.columns:
    sns.violinplot(x=data_show[col])
    plt.show()
```

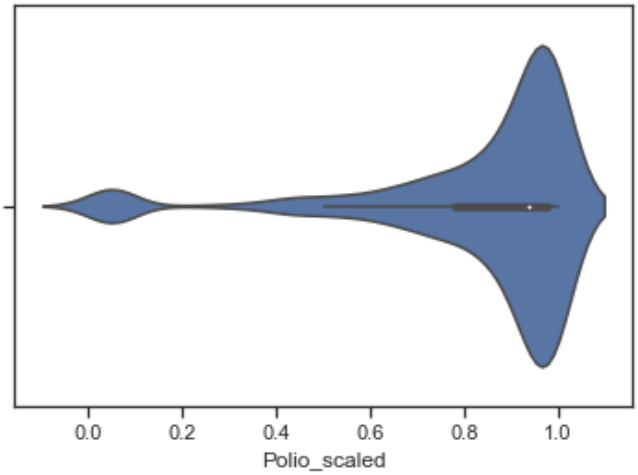


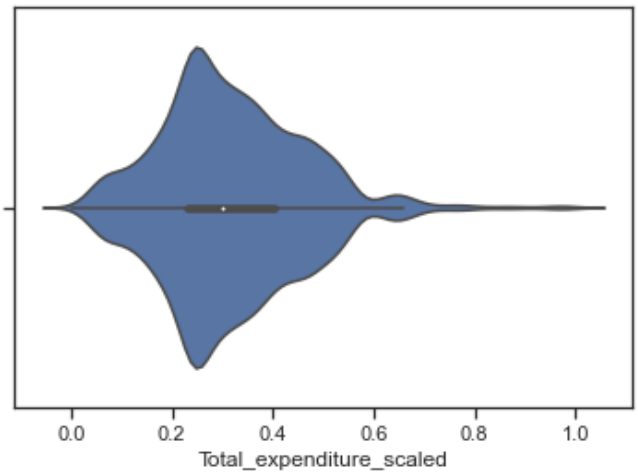


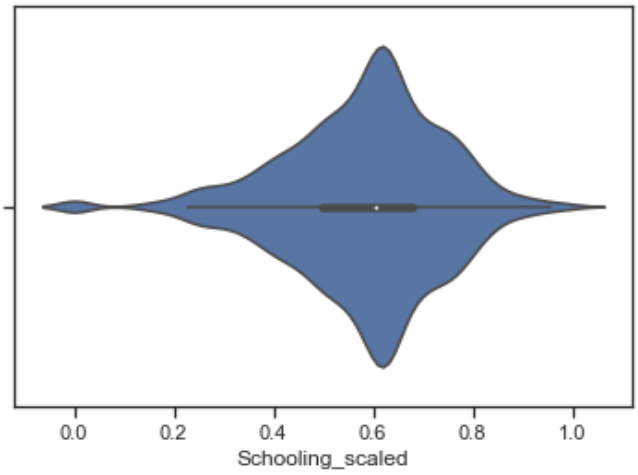












Выбор метрик для последующей оценки качества моделей.

В качестве метрик для решения задачи регрессии будем использовать следующие метрики:

Mean absolute error - средняя абсолютная ошибка

$MAE(y, \hat{y}) = \frac{1}{N} \cdot \sum_{i=1}^{N} |y_i - \hat{y}_i|$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки

- N - размер тестовой выборки

Чем ближе значение к нулю, тем лучше качество регрессии.

Основная проблема метрики состоит в том, что она не нормирована.

Вычисляется с помощью функции **mean_absolute_error.**

Mean squared error- средняя квадратичная ошибка

$$MSE(y, \hat{y}) = \frac{1}{N} \cdot \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки

Вычисляется с помощью функции **mean_squared_error.**

Median absolute error

$$MedAE(y, \hat{y}) = median(|y_1 - \hat{y_1}|, \ldots, |y_N - \hat{y_N}|)$$

Метрика интересна тем, что является устойчивой к выбросам в данных.

Вычисляется с помощью функции **median_absolute_error.**

Метрика R^2 или коэффициент детерминации

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \overline{y_i})^2}$$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки
- $\overline{y_i} = \frac{1}{N} \cdot \sum_{i=1}^{N} y_i$

Вычисляется с помощью функции **r2_score.**

Сохранение и визуализация метрик

Класс, который позволит сохранять метрики качества построенных моделей и реализует визуализацию метрик качества.

In [329]:

```
class MetricLogger:

    def __init__(self):
        self.df = pd.DataFrame(
```

```
            {'metric': pd.Series([], dtype='str'),
            'alg': pd.Series([], dtype='str'),
            'value': pd.Series([], dtype='float')})

    def add(self, metric, alg, value):
        """
        Добавление значения
        """
        # Удаление значения если оно уже было ранее добавлено
        self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)].index, i
nplace = True)
        # Добавление нового значения
        temp = [{'metric':metric, 'alg':alg, 'value':value}]
        self.df = self.df.append(temp, ignore_index=True)

    def get_data_for_metric(self, metric, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        temp_data = self.df[self.df['metric']==metric]
        temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
        return temp_data_2['alg'].values, temp_data_2['value'].values

    def plot(self, str_header, metric, ascending=True, figsize=(5, 5)):
        """
        Вывод графика
        """
        array_labels, array_metric = self.get_data_for_metric(metric, ascending)
        fig, ax1 = plt.subplots(figsize=figsize)
        pos = np.arange(len(array_metric))
        rects = ax1.barh(pos, array_metric,
                         align='center',
                         height=0.5,
                         tick_label=array_labels)
        ax1.set_title(str_header)
        for a,b in zip(pos, array_metric):
            plt.text(0.003, a-0.003, str(round(b,3)), color='white')
        plt.show()
```

Выбор наиболее подходящих моделей для решения задачи классификации или регрессии.

Для задачи регрессии будем использовать следующие модели:

- Линейная регрессия
- Метод ближайших соседей
- Машина опорных векторов
- Решающее дерево
- Случайный лес
- Градиентный бустинг

Формирование обучающей и тестовой выборок на основе исходного набора данных.

In [330]:

```
# Признаки для задачи регресии
regr_cols = ['percentage_expenditure_scaled','BMI_scaled','Diphtheria_scaled',
             'Polio_scaled', 'Total_expenditure_scaled','Schooling_scaled']
```

In [331]:

```
X = data[regr_cols]
Y = data.Life_expectancy_scaled
print('Входные данные:\n\n', X.head(), '\n\nВыходные данные:\n\n', Y.head())
```

```
Входные данные:
```

```
    percentage_expenditure_scaled  BMI_scaled  Diphtheria_scaled  Polio_scaled  \
0                        0.003659    0.209733           0.649485      0.031250
1                        0.003774    0.203940           0.618557      0.572917
2                        0.003759    0.198146           0.639175      0.614583
3                        0.004014    0.192352           0.670103      0.666667
4                        0.000364    0.187717           0.680412      0.677083

   Total_expenditure_scaled  Schooling_scaled
0                  0.452118          0.487923
1                  0.453279          0.483092
2                  0.450377          0.478261
3                  0.473012          0.473430
4                  0.435287          0.458937

Выходные данные:

 0    0.544592
1    0.447818
2    0.447818
3    0.440228
4    0.434535
Name: Life_expectancy_scaled, dtype: float64
```

In [332]:

```
X_train,  X_test,  Y_train,  Y_test = train_test_split(X,  Y, random_state = 0, test_siz
e = 0.1)
print('Входные параметры обучающей выборки:\n\n',X_train.head(), \
      '\n\nВходные параметры тестовой выборки:\n\n', X_test.head(), \
      '\n\nВыходные параметры обучающей выборки:\n\n', Y_train.head(), \
      '\n\nВыходные параметры тестовой выборки:\n\n', Y_test.head())
```

```
Входные параметры обучающей выборки:

       percentage_expenditure_scaled  BMI_scaled  Diphtheria_scaled  \
1059                        0.033754    0.566628           0.731959
861                         0.000527    0.146002           0.989691
847                         0.012053    0.209733           0.350515
890                         0.001564    0.156431           0.030928
485                         0.005180    0.289687           0.845361

      Polio_scaled  Total_expenditure_scaled  Schooling_scaled
1059      0.645833                  0.338363          0.516908
861       0.989583                  0.160766          0.241546
847       0.427083                  0.123622          0.396135
890       0.604167                  0.250725          0.342995
485       0.833333                  0.284968          0.468599

Входные параметры тестовой выборки:

       percentage_expenditure_scaled  BMI_scaled  Diphtheria_scaled  \
867                         0.138608    0.676709           0.938144
1780                        0.000000    0.264195           0.896907
621                         0.000000    0.015064           0.402062
2715                        0.000727    0.198146           0.783505
2717                        0.004768    0.187717           0.783505

      Polio_scaled  Total_expenditure_scaled  Schooling_scaled
867       0.937500                  0.348810          0.797101
1780      0.895833                  0.245502          0.439614
621       0.395833                  0.120720          0.458937
2715      0.822917                  0.397562          0.483092
2717      0.822917                  0.418456          0.483092

Выходные параметры обучающей выборки:

 1059    0.671727
861     0.432638
847     0.326376
890     0.421252
485     0.360531
Name: Life expectancy scaled, dtype: float64
```

```
Выходные параметры тестовой выборки:

 867     0.777989
1780    0.574953
621     0.309298
2715    0.478178
2717    0.449715
Name: Life_expectancy_scaled, dtype: float64
```

In [333]:

```
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

Out[333]:

```
((2644, 6), (294, 6), (2644,), (294,))
```

Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.

In [351]:

```
# Модели
regr_models = {'LR': LinearRegression(),
               'KNN_5':KNeighborsRegressor(n_neighbors=10),
               'SVR':SVR(),
               'Tree':DecisionTreeRegressor(),
               'RF':RandomForestRegressor(),
               'GB':GradientBoostingRegressor()}
```

In [352]:

```
# Сохранение метрик
regrMetricLogger = MetricLogger()
```

In [355]:

```
def regr_train_model(model_name, model, regrMetricLogger):
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)

    mae = mean_absolute_error(Y_test, Y_pred)
    mse = mean_squared_error(Y_test, Y_pred)
    medae = median_absolute_error(Y_test, Y_pred)
    r2 = r2_score(Y_test, Y_pred)

    regrMetricLogger.add('MAE', model_name, mae)
    regrMetricLogger.add('MSE', model_name, mse)
    regrMetricLogger.add('MedAE', model_name, medae)
    regrMetricLogger.add('R2', model_name, r2)

    print('{} \t MAE={}, MSE={}, MedAE={}, R2={}'.format(
        model_name, round(mae, 3), round(mse, 3), round(medae, 3), round(r2, 3)))
```

In [356]:

```
for model_name, model in regr_models.items():
    regr_train_model(model_name, model, regrMetricLogger)
```

```
LR   MAE=0.079, MSE=0.011, MedAE=0.064, R2=0.667
KNN_5   MAE=0.061, MSE=0.007, MedAE=0.046, R2=0.781
SVR   MAE=0.07, MSE=0.008, MedAE=0.06, R2=0.755
Tree   MAE=0.073, MSE=0.011, MedAE=0.044, R2=0.641
RF   MAE=0.048, MSE=0.004, MedAE=0.036, R2=0.866
GB   MAE=0.064, MSE=0.007, MedAE=0.051, R2=0.785
```

Подбор гиперпараметров для выбранных моделей.

In [357]:

```
n_range = np.array(range(5,55,5))
tuned_parameters = [{'n_neighbors': n_range}]
tuned_parameters
```

Out[357]:

```
[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}]
```

In [358]:

```
%%time
regr_gs = GridSearchCV(KNeighborsRegressor(), tuned_parameters, cv=5, scoring='neg_mean_
squared_error')
regr_gs.fit(X_train, Y_train)
```

```
Wall time: 880 ms
```

Out[358]:

```
GridSearchCV(cv=5, estimator=KNeighborsRegressor(),
             param_grid=[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}
],
             scoring='neg_mean_squared_error')
```

In [359]:

```
# Лучшая модель
regr_gs.best_estimator_
```

Out[359]:

```
KNeighborsRegressor()
```

In [360]:

```
# Лучшее значение параметров
regr_gs.best_params_
```

Out[360]:

```
{'n_neighbors': 5}
```

In [361]:

```
regr_gs_best_params_txt = str(regr_gs.best_params_['n_neighbors'])
regr_gs_best_params_txt
```

Out[361]:

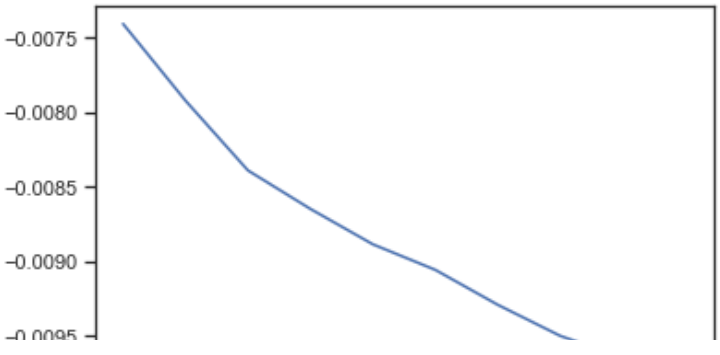
```
'5'
```

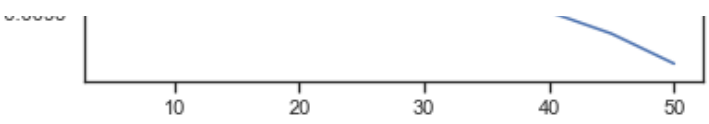
In [362]:

```
# Изменение качества на тестовой выборке в зависимости от К-соседей
plt.plot(n_range, regr_gs.cv_results_['mean_test_score'])
```

Out[362]:

```
[<matplotlib.lines.Line2D at 0x16cf4b61850>]
```





Повторение пункта **8** для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством **baseline**-моделей.

In [363]:

```
regr_models_grid = {'KNN_10':KNeighborsRegressor(n_neighbors=10),
                    str('KNN_'+regr_gs_best_params_txt):regr_gs.best_estimator_}
```

In [364]:

```
for model_name, model in regr_models_grid.items():
    regr_train_model(model_name, model, regrMetricLogger)
```

```
KNN_10   MAE=0.061, MSE=0.007, MedAE=0.046, R2=0.781
KNN_5   MAE=0.059, MSE=0.007, MedAE=0.047, R2=0.792
```

Формирование выводов о качестве построенных моделей на основе выбранных метрик.

In [365]:

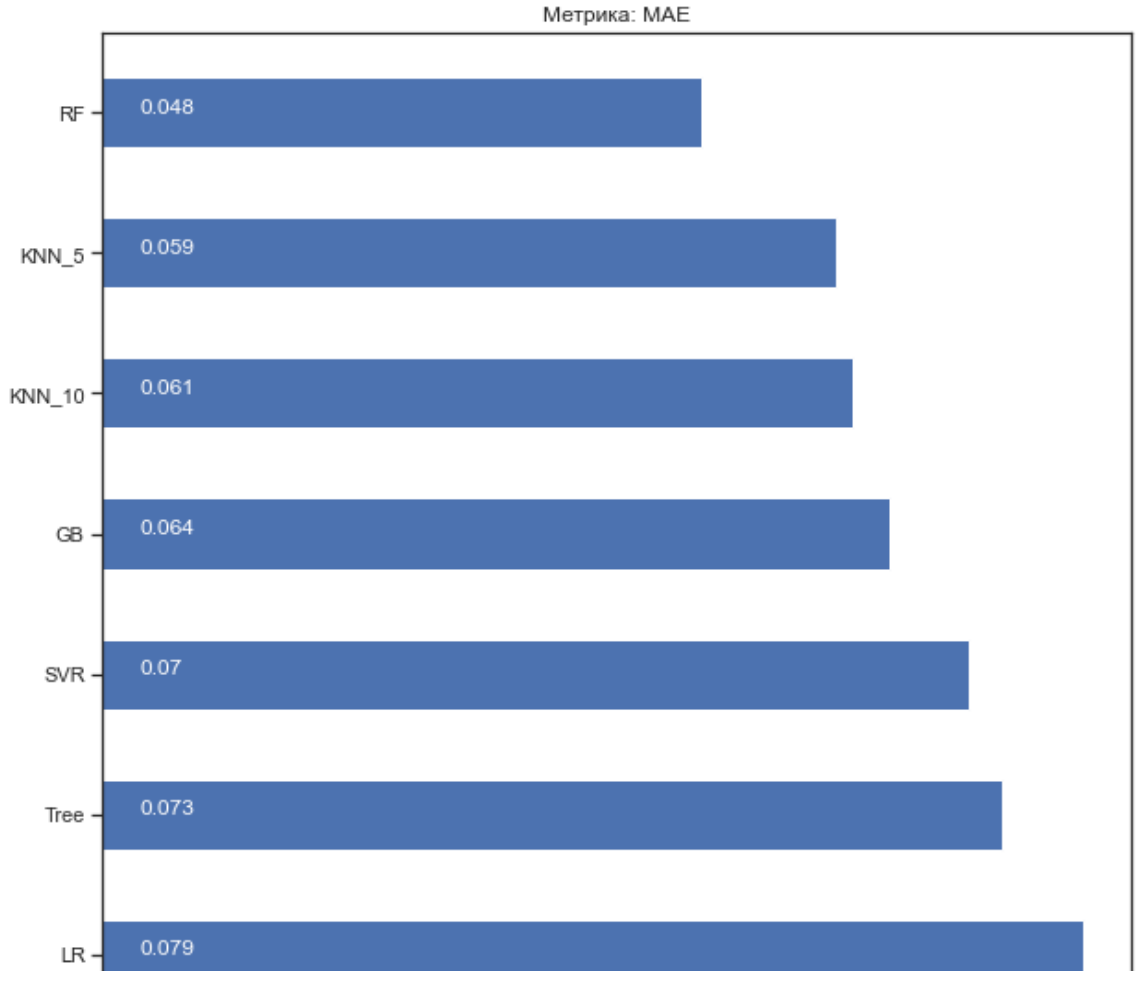
```
# Метрики качества модели
regr_metrics = regrMetricLogger.df['metric'].unique()
regr_metrics
```

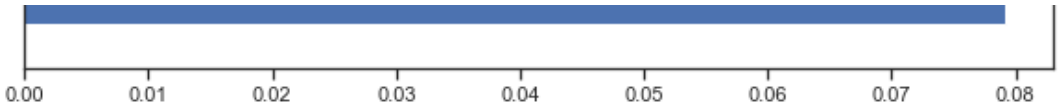
Out[365]:

```
array(['MAE', 'MSE', 'MedAE', 'R2'], dtype=object)
```

In [366]:

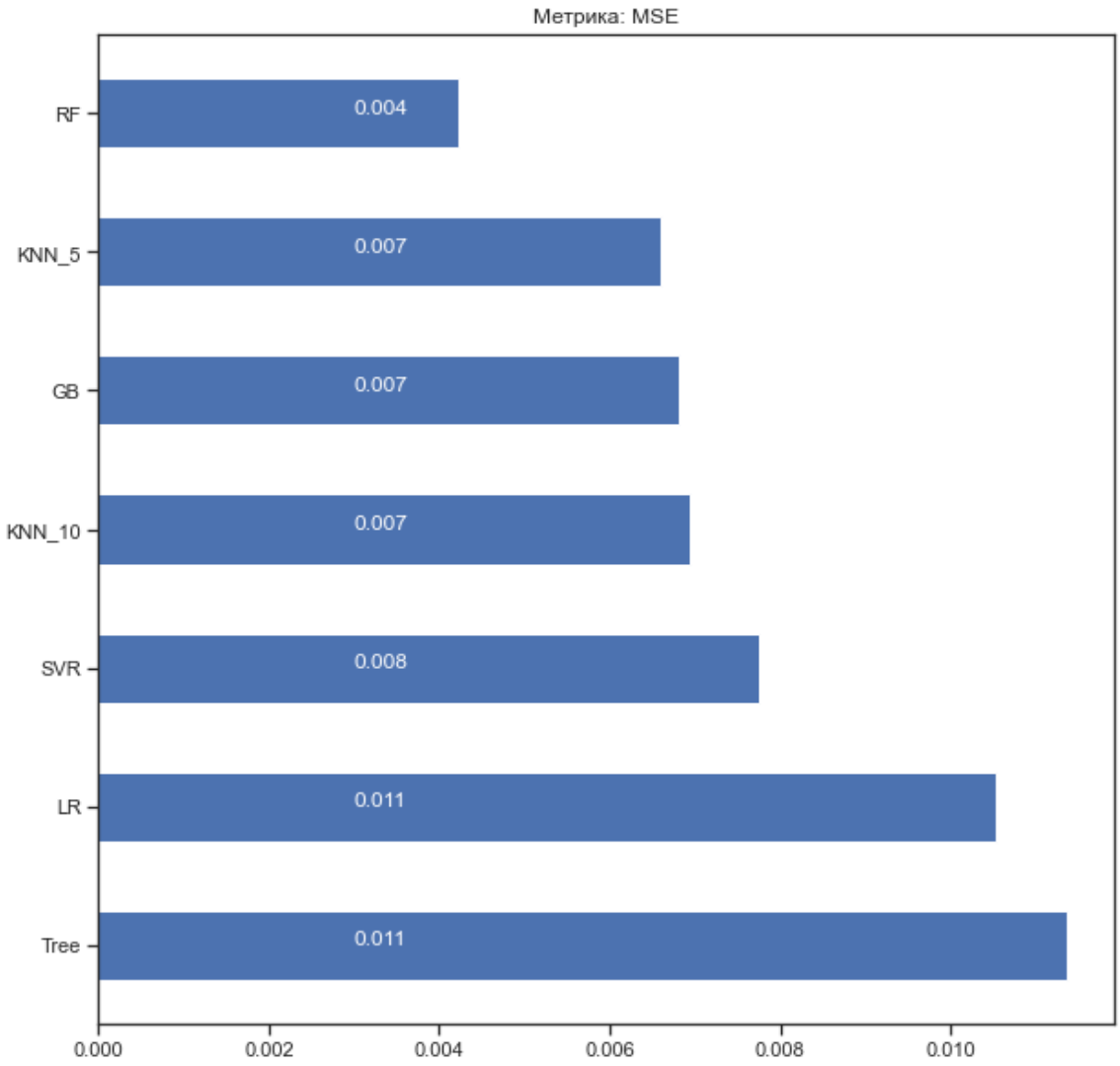
```
regrMetricLogger.plot('Метрика: ' + 'MAE', 'MAE', ascending=False, figsize=(10, 10))
```





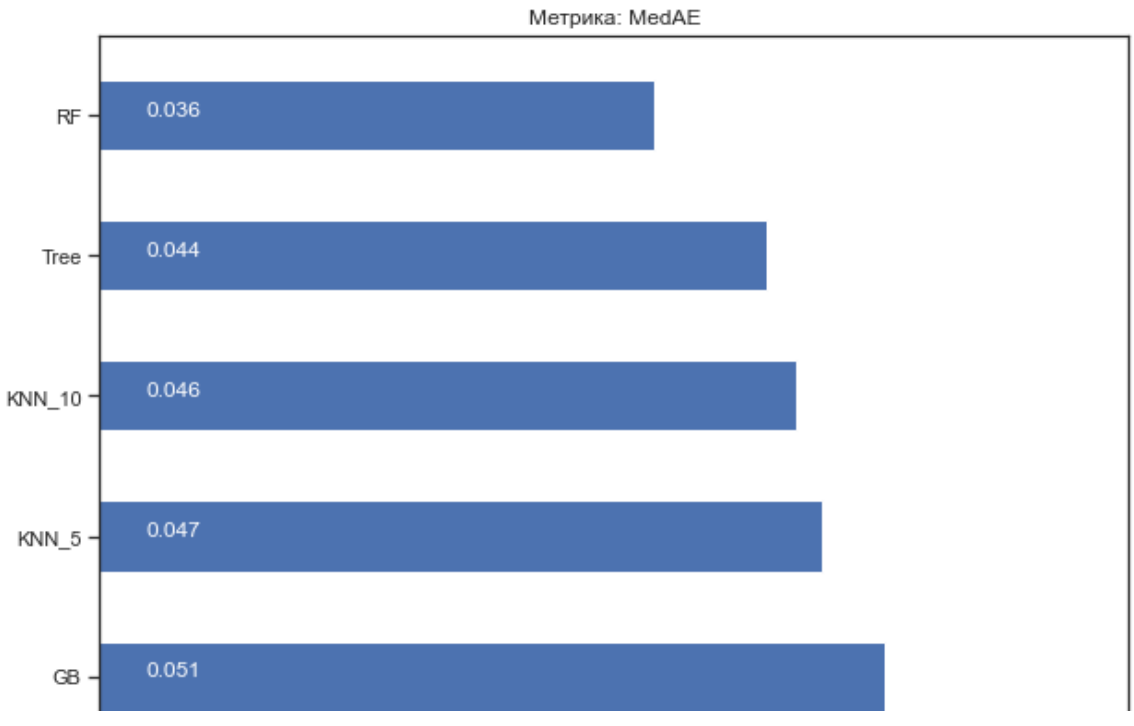
In [367]:

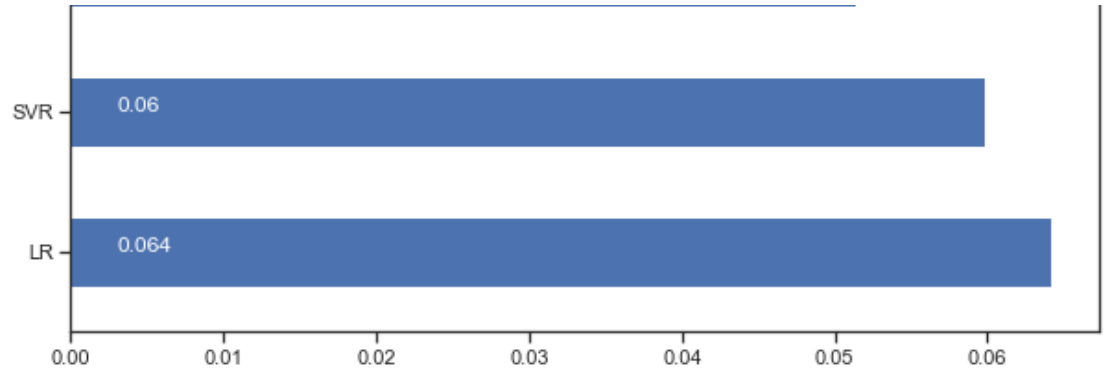
```
regrMetricLogger.plot('Метрика: ' + 'MSE', 'MSE', ascending=False, figsize=(10,10))
```



In [368]:

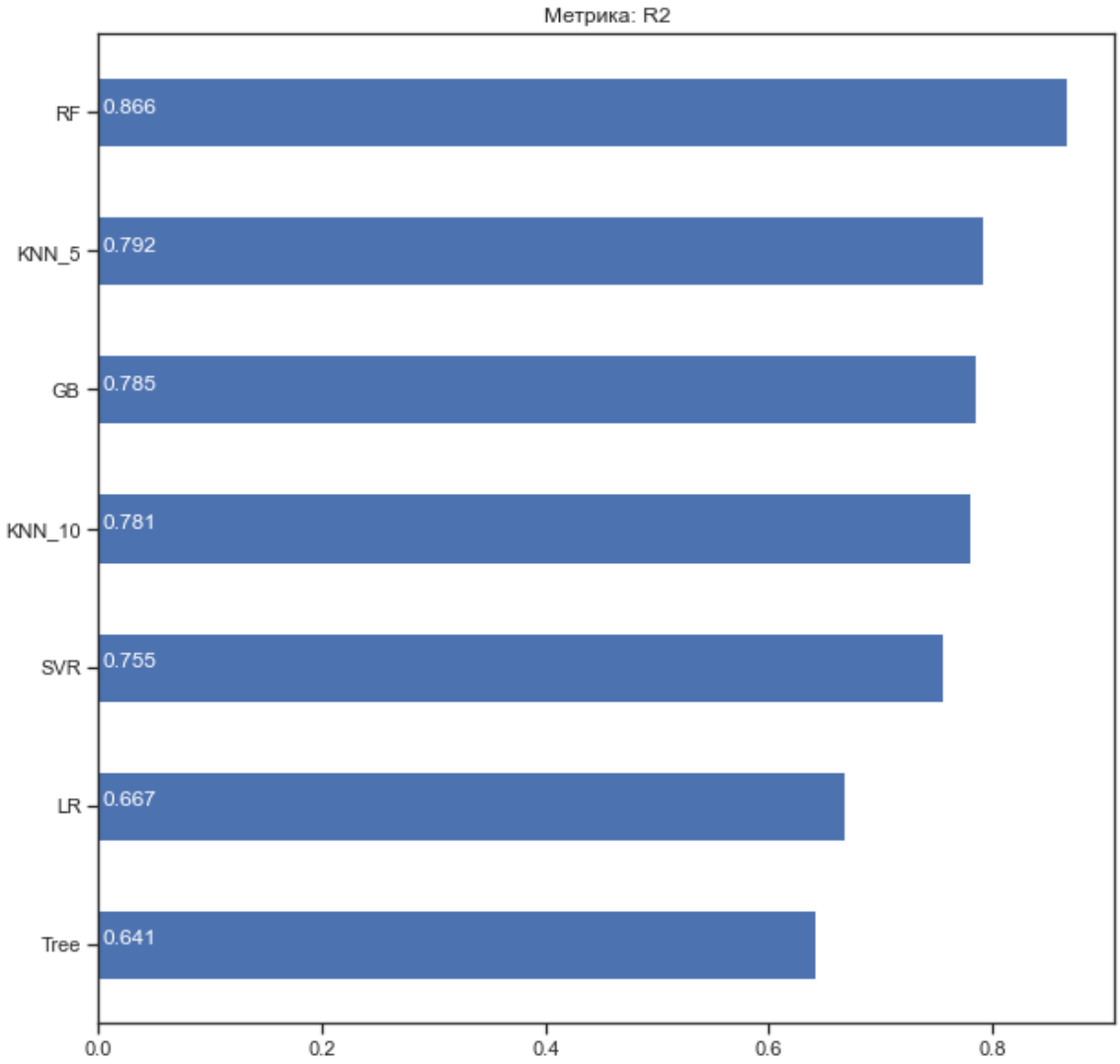
```
regrMetricLogger.plot('Метрика: ' + 'MedAE', 'MedAE', ascending=False, figsize=(10, 10))
```





In [369]:

```
regrMetricLogger.plot('Метрика: ' + 'R2', 'R2', ascending=True, figsize=(10, 10))
```



Вывод: лучшей оказалась модель на основе случайного леса.

In [ ]:

```
```

2.1.Текст программы веб-приложения, предназначенного для анализа данных:

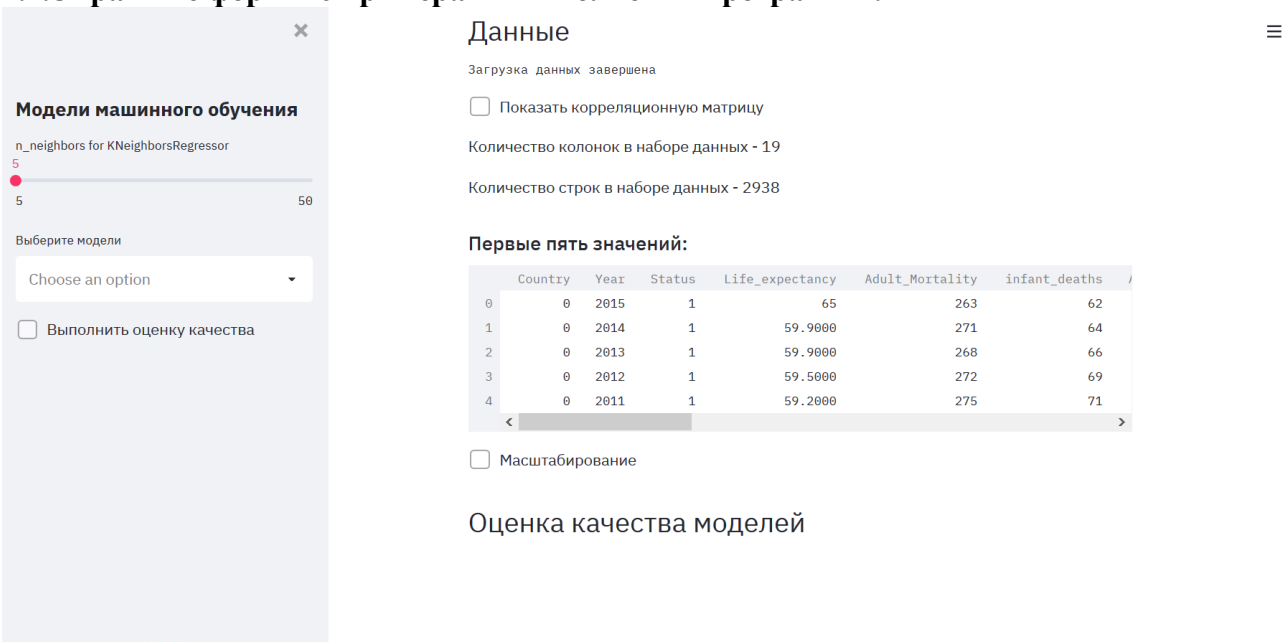
```
import streamlit as st
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import  KNeighborsRegressor
from sklearn.tree import  DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolut
e_error,r2_score
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
import matplotlib.pyplot as plt


def load_data():
    '''
    Загрузка данных
    '''
    data = pd.read_csv('C:\Py\Lab6_TMO\data\Life Expectancy Data.csv')
    return data

st.sidebar.header('Модели машинного обучения')
n_neighbors = st.sidebar.slider("n_neighbors for KNeighborsRegressor", 5, 50, value
=5, step = 5)

st.header('Данные')
status = st.text('Загрузка данных ...')
data = load_data()
status.text('Загрузка данных завершена')

# Удаление колонок со слишком большим количеством пропусков
data.drop(['Hepatitis_B','GDP','Population',], axis = 1, inplace = True)

# Обработка пропусков
imp_num = SimpleImputer(strategy='median')
imp_num2 = SimpleImputer(strategy='most_frequent')
data[['Life_expectancy']] = imp_num2.fit_transform(data[['Life_expectancy']])
data[['Adult_Mortality']] = imp_num.fit_transform(data[['Adult_Mortality']])
data[['Alcohol']] = imp_num.fit_transform(data[['Alcohol']])
data[['BMI']] = imp_num.fit_transform(data[['BMI']])
data[['Polio']] = imp_num.fit_transform(data[['Polio']])
data[['Total_expenditure']] = imp_num2.fit_transform(data[['Total_expenditure']])
data[['Diphtheria']] = imp_num.fit_transform(data[['Diphtheria']])
```

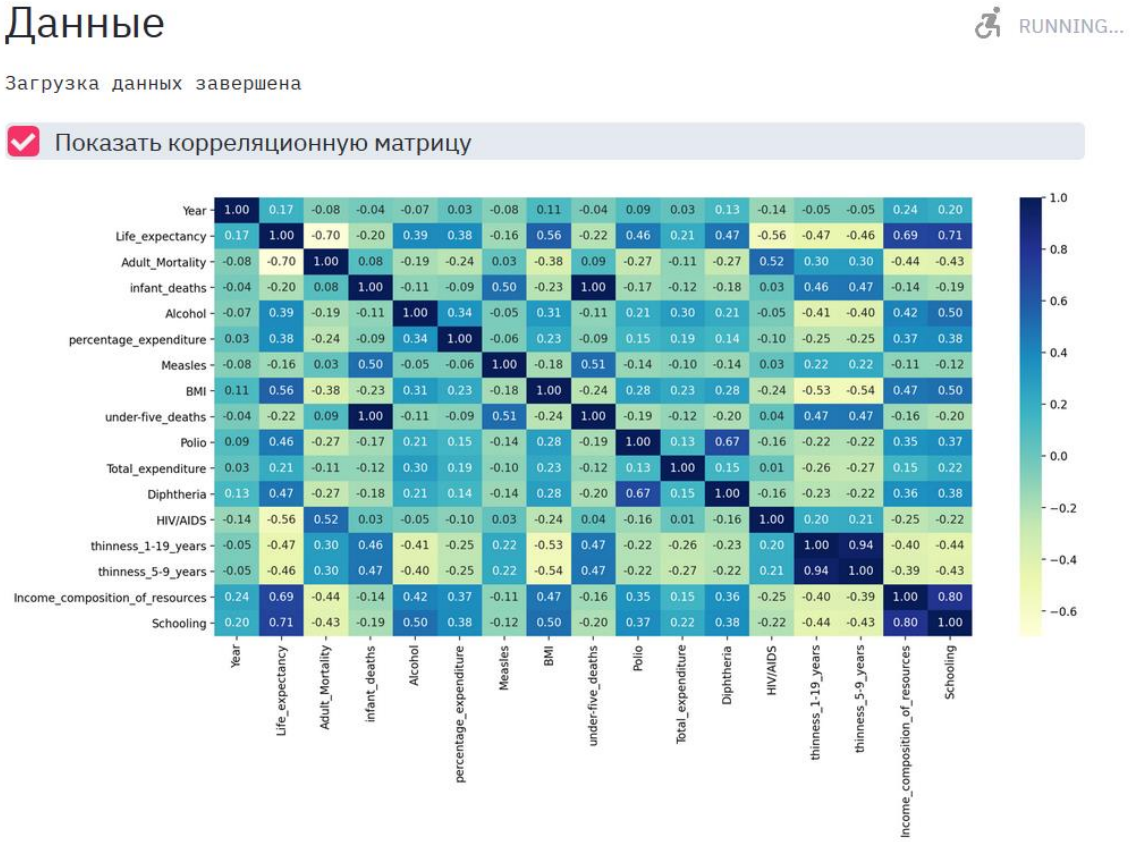
```
data[['thinness_1-19_years']] = imp_num.fit_transform(data[['thinness_1-
19_years']])
data[['thinness_5-9_years']] = imp_num.fit_transform(data[['thinness_5-9_years']])
data[['Income_composition_of_resources']] = imp_num.fit_transform(data[['Income_com
position_of_resources']])
data[['Schooling']] = imp_num2.fit_transform(data[['Schooling']])

if st.checkbox('Показать корреляционную матрицу'):
    fig1, ax = plt.subplots(figsize=(15,7))
    sns.heatmap(data.corr(), annot=True,cmap='YlGnBu', fmt='.2f')
    st.pyplot(fig1)

data_len = data.shape[0]
st.write('Количество колонок в наборе данных - {}'.format(data.shape[1]))
st.write('Количество строк в наборе данных - {}'.format(data_len))

# Кодирование категориальных признаков целочисленными значениями
le = LabelEncoder()
le.fit(data.Status)
data.Status = le.transform(data.Status)
le.fit(data.Country)
data.Country = le.transform(data.Country)

st.subheader('Первые пять значений:')
st.write(data.head())

scale_cols = ['Life_expectancy', 'Adult_Mortality', 'infant_deaths', 'percentage_ex
penditure', 'Measles',
              'BMI','under-
five_deaths','Polio','Total_expenditure','Diphtheria','thinness_1-19_years',
             'thinness_5-9_years','Schooling']

if st.checkbox('Масштабирование'):
    sc1 = MinMaxScaler()
    data[scale_cols] = sc1.fit_transform(data[scale_cols])
    st.write(data.head())

st.header('Оценка качества моделей')


def preprocess_data(data):
    regr_cols = ['percentage_expenditure','BMI','Diphtheria',
        'Polio', 'Total_expenditure','Schooling']
    X = data[regr_cols]
    Y = data.Life_expectancy
    X_train,  X_test,  Y_train,  Y_test = train_test_split(X,  Y, random_state = 0,
 test_size = 0.1)
    return X_train, X_test, Y_train, Y_test

X_train, X_test, Y_train, Y_test = preprocess_data(data)
```

```
LR = LinearRegression()
LR.fit(X_train, Y_train)

KNN = KNeighborsRegressor(n_neighbors=n_neighbors)
KNN.fit(X_train, Y_train)

SVR = SVR()
SVR.fit(X_train, Y_train)

Tree = DecisionTreeRegressor()
Tree.fit(X_train, Y_train)

RF = RandomForestRegressor()
RF.fit(X_train, Y_train)

GB = GradientBoostingRegressor()
GB.fit(X_train, Y_train)

metrics = [r2_score, mean_absolute_error, mean_squared_error, median_absolute_error
]

models_list = [LR, KNN, SVR, Tree, RF, GB]
model_names = [i.__class__.__name__ for i in models_list]
models = st.sidebar.multiselect("Выберите модели", model_names)

current_models_list = []
for i in models:
    for j in models_list:
        if i == j.__class__.__name__:
            current_models_list.append(j)

if st.sidebar.checkbox('Выполнить оценку качества'):
    for name in metrics:
        st.subheader(name.__name__)

        array_labels = []
        array_metric = []

        for func in current_models_list:
            Y_pred = func.predict(X_test)

            array_labels.append(func.__class__.__name__)
            array_metric.append(name(Y_pred, Y_test))

            st.text("{} - {}".format(func.__class__.__name__, name(Y_pred, Y_test))
)

        fig, ax1 = plt.subplots(figsize=(3,3))
        pos = np.arange(len(array_metric))
        rects = ax1.barh(
            pos,
```

```
        array_metric,
        align="center",
        height=0.5,
        tick_label=array_labels,
    )
    for a, b in zip(pos, array_metric):
        plt.text(0, a - 0.1, str(round(b, 3)), color="white")
    st.pyplot(fig)
```

2.2.Экранные формы с примерами выполнения программы:





Модели машинного обучения

n_neighbors for KNeighborsRegressor

5

5 50

Выберите модели

Choose an option

Выполнить оценку качества

Данные

Загрузка данных завершена

Показать корреляционную матрицу

Количество колонок в наборе данных - 19

Количество строк в наборе данных - 2938

Первые пять значений:

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths
0	0	2015	1	65	263	62
1	0	2014	1	59.9000	271	64
2	0	2013	1	59.9000	268	66
3	0	2012	1	59.5000	272	69
4	0	2011	1	59.2000	275	71

Масштабирование

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths
0	0	2015	1	0.5446	0.3629	0.0344
1	0	2014	1	0.4478	0.3740	0.0356
2	0	2013	1	0.4478	0.3698	0.0367
3	0	2012	1	0.4402	0.3753	0.0383
4	0	2011	1	0.4345	0.3795	0.0394

Модели машинного обучения

n_neighbors for KNeighborsRegressor

5

5 50

Выберите модели

LinearRegression SVR KNeighborsRegressor RandomForestRegres... DecisionTreeRegressor GradientBoostingReg...

Выполнить оценку качества

r2_score

LinearRegression - 0.4390970048079955

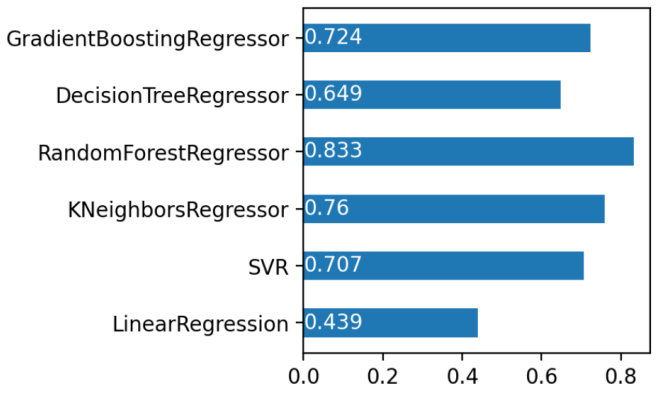
SVR - 0.706815582473792

KNeighborsRegressor - 0.7598411660615659

RandomForestRegressor - 0.8330138689913432

DecisionTreeRegressor - 0.6490319530521445

GradientBoostingRegressor - 0.7238959839319483



mean_absolute_error

LinearRegression - 0.07914244592187206

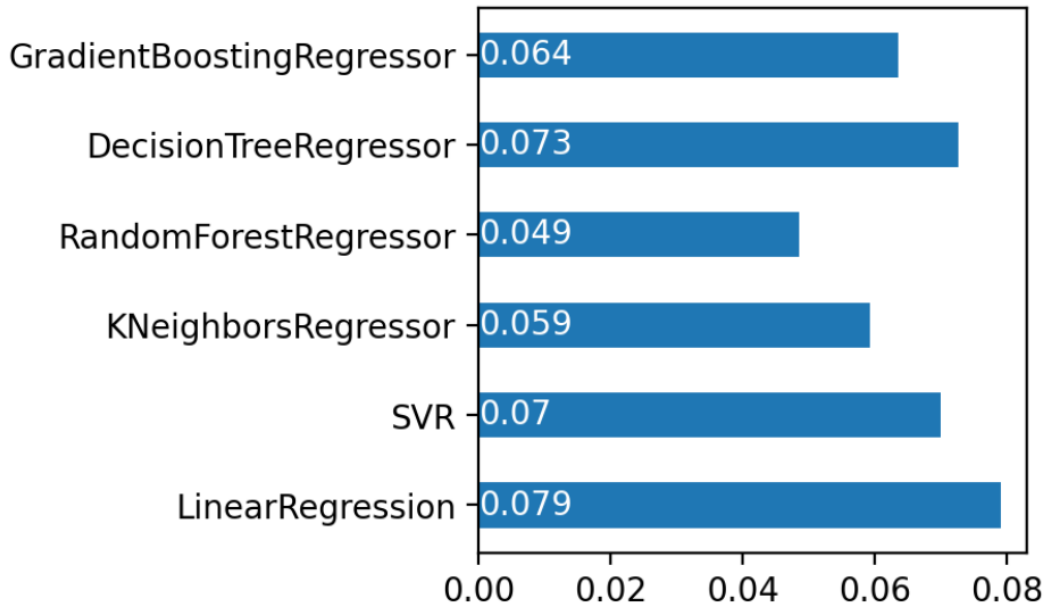
SVR - 0.06994406571138588

KNeighborsRegressor - 0.05924176122061727

RandomForestRegressor - 0.048582536957435286

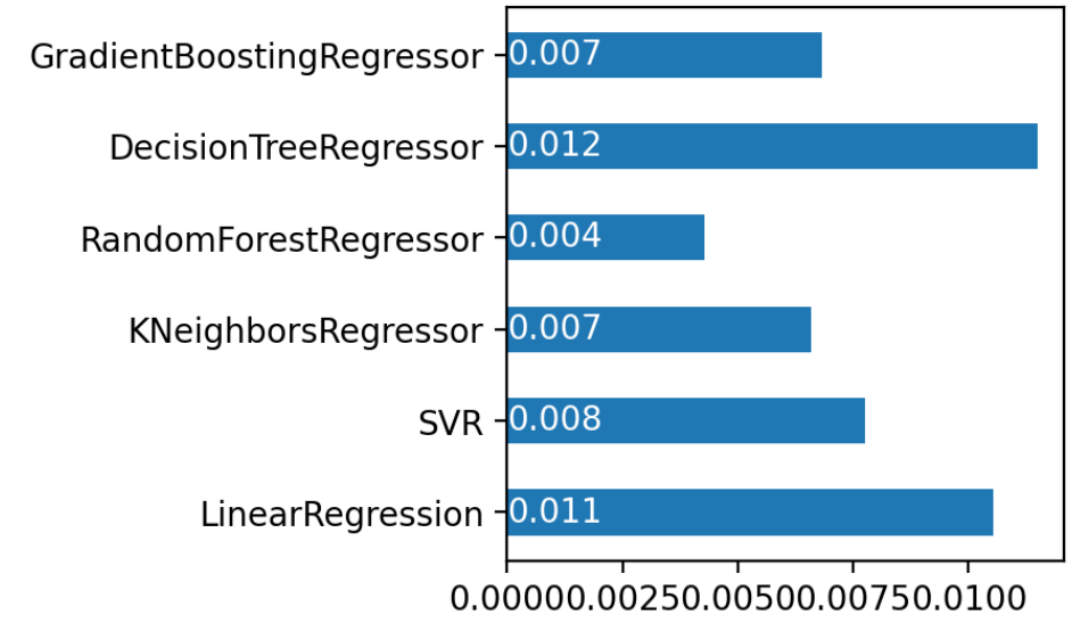
DecisionTreeRegressor - 0.07276458970685047

GradientBoostingRegressor - 0.0635846081031575



mean_squared_error

```
LinearRegression - 0.01052658442207902

SVR - 0.007753292240311394

KNeighborsRegressor - 0.006593115302067451

RandomForestRegressor - 0.004282311461587288

DecisionTreeRegressor - 0.011500199026785837

GradientBoostingRegressor - 0.006809951103329997
```



Заключение

В ходе выполнения курсовой работы было выполнено решение задачи машинного обучения на основе материалов дисциплины.

Целевым признаком задачи регрессии был выбран признак «Life_expectancy». Было построено шесть моделей:

- Линейная регрессия
- Метод ближайших соседей
- Машина опорных векторов
- Решающее дерево
- Случайный лес
- Градиентный бустинг

В результате выполнения работы лучшее качество показала модель «Случайный лес».

Список использованных источников информации

- Конспект лекций курса «Технологии машинного обучения»: https://github.com/ugapanyuk/ml_course_2021/wiki/COURSE_TMO

- Набор данных Life Expectancy (WHO) https://www.kaggle.com/kumarajarshi/life-expectancy-who