Московский государственный технический университет им. Н.Э.

Баумана Кафедра «Системы обработки информации и управления»

Лабораторная работа №1

по дисциплине

«Методы машинного обучения»

на тему

«Обработка признаков(часть 2)»

Выполнил:

студент группы ИУ5И-22М

Чжан Аньци

Москва — 2022 г.

1. Цель лабораторной работы

изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

2. Задание

Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

a) масштабирование признаков (не менее чем тремя способами);
b) обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
c) обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);
d) отбор признаков:
    i. один метод из группы методов фильтрации (filter methods);
    ii. один метод из группы методов обертывания (wrapper methods);
    iii. один метод из группы методов вложений (embedded methods).

3. Ход выполнения работы

Импортируйте необходимую библиотеку и загрузите набор данных

In [13]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn import preprocessing
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
import warnings
warnings.filterwarnings("ignore")
```

In [14]:
```
dataset=pd.read_csv('Life Expectancy Data.csv')
```

In [15]:
```
dataset.head()
```

Out[15]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GD
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1	584.2592
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1	612.6965
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.7449
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	0.1	669.9590
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1	63.5372

5 rows × 22 columns

3.1 масштабирование признаков

Выберите данные для использования

In [17]:
```
data=dataset[['Alcohol','Measles ','Polio','GDP']]
data
```

Out[17]:

	Alcohol	Measles	Polio	GDP
0	0.01	1154	6.0	584.259210
1	0.01	492	58.0	612.696514
2	0.01	430	62.0	631.744976
3	0.01	2787	67.0	669.959000
4	0.01	3013	68.0	63.537231
...	...	...	...	...
2933	4.36	31	67.0	454.366654
2934	4.06	998	7.0	453.351155
2935	4.43	304	73.0	57.348340
2936	1.72	529	76.0	548.587312
2937	1.68	1483	78.0	547.358878

2938 rows × 4 columns

3.1.1 Стандартный масштабатор(StandardScaler)

In [18]:
```
def arr_df(a):
    df = pd.DataFrame(a, columns=data.columns)
    return df
scaler1 = StandardScaler()
scaled_1 = arr_df(scaler1.fit_transform(data))
scaled_1
```

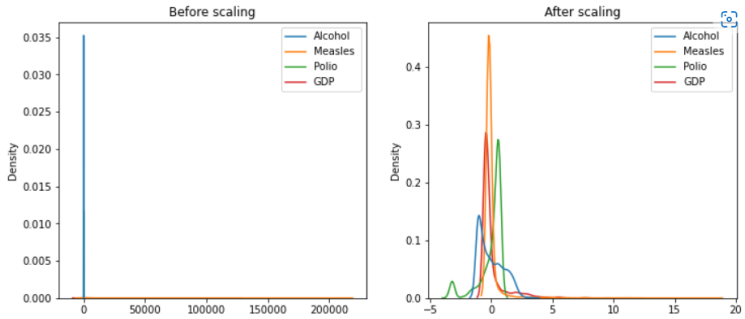
Out[18]:

	Alcohol	Measles	Polio	GDP
0	-1.133571	-0.110384	-3.268019	-0.483546
1	-1.133571	-0.168124	-1.048077	-0.481553
2	-1.133571	-0.173531	-0.877312	-0.480218
3	-1.133571	0.032045	-0.663856	-0.477539
4	-1.133571	0.051757	-0.621165	-0.520044
...	...	...	...	...
2933	-0.059941	-0.208332	-0.663856	-0.492650
2934	-0.133984	-0.123991	-3.225328	-0.492722
2935	-0.042664	-0.184521	-0.407709	-0.520477
2936	-0.711523	-0.164897	-0.279635	-0.486046
2937	-0.721396	-0.081689	-0.194253	-0.486132

2938 rows × 4 columns

In [19]:
```
def data_visualize(columns, df1, df2, label1, label2):
  fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 5))
  ax1.set_title(label1)
  sns.kdeplot(data=df1[columns], ax=ax1)
  ax2.set_title(label2)
  sns.kdeplot(data=df2[columns], ax=ax2)
  plt.show()

data_visualize(data.columns, data, scaled_1, 'Before scaling', 'After scaling')
```



3.1.2 МинМаксСкалер(MinMaxScaler)

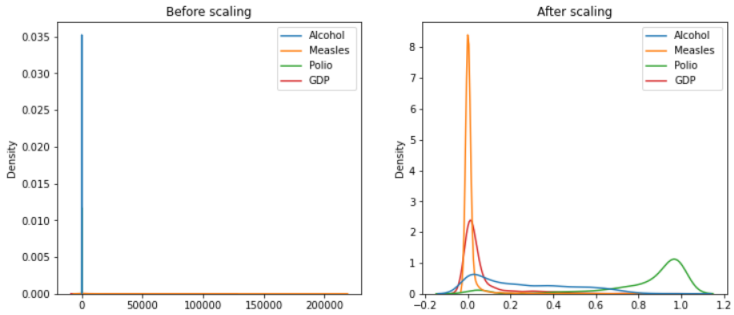
In [20]:
```
scaler2 = MinMaxScaler()
scaled_2 = arr_df(scaler2.fit_transform(data))
scaled_2
```

Out[20]:

	Alcohol	Measles	Polio	GDP
0	0.000000	0.005439	0.031250	0.004889
1	0.000000	0.002319	0.572917	0.005127
2	0.000000	0.002027	0.614583	0.005287
3	0.000000	0.013135	0.666667	0.005608
4	0.000000	0.014200	0.677083	0.000519
...	...	...	...	...
2933	0.243561	0.000146	0.666667	0.003799
2934	0.226764	0.004703	0.041667	0.003790
2935	0.247480	0.001433	0.729167	0.000467
2936	0.095745	0.002493	0.760417	0.004589
2937	0.093505	0.006989	0.781250	0.004579

2938 rows × 4 columns

In [21]:
```
data_visualize(data.columns, data, scaled_2, 'Before scaling', 'After scaling')
```



3.1.3 RobustScaler

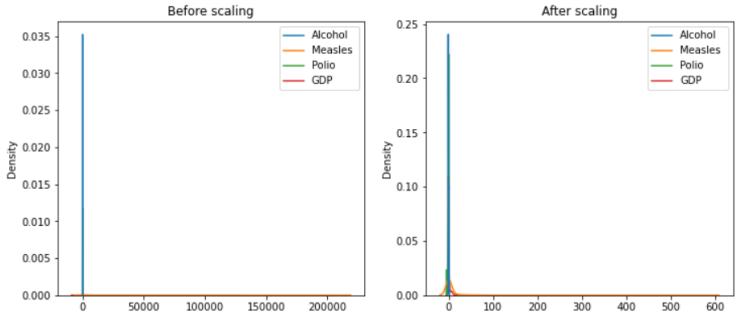
In [22]:
```
scaler3 = RobustScaler()
scaled_3 = arr_df(scaler3.fit_transform(data))
scaled_3
```

Out[22]:

	Alcohol	Measles	Polio	GDP
0	-0.548718	3.156142	-4.578947	-0.217132
1	-0.548718	1.318529	-1.842105	-0.211911
2	-0.548718	1.146426	-1.631579	-0.208414
3	-0.548718	7.689105	-1.368421	-0.201398
4	-0.548718	8.316447	-1.315789	-0.312732
...	...	...	...	...
2933	0.088645	0.038862	-1.368421	-0.240979
2934	0.044689	2.723109	-4.526316	-0.241165
2935	0.098901	0.796669	-1.052632	-0.313868
2936	-0.298168	1.421235	-0.894737	-0.223681
2937	-0.304029	4.069396	-0.789474	-0.223906

2938 rows × 4 columns

In [23]:
```
data_visualize(data.columns, data, scaled_3, 'Before scaling', 'After scaling')
```



3.2 обработку выбросов для числовых признаков

3.2.1 Удаление выборсов

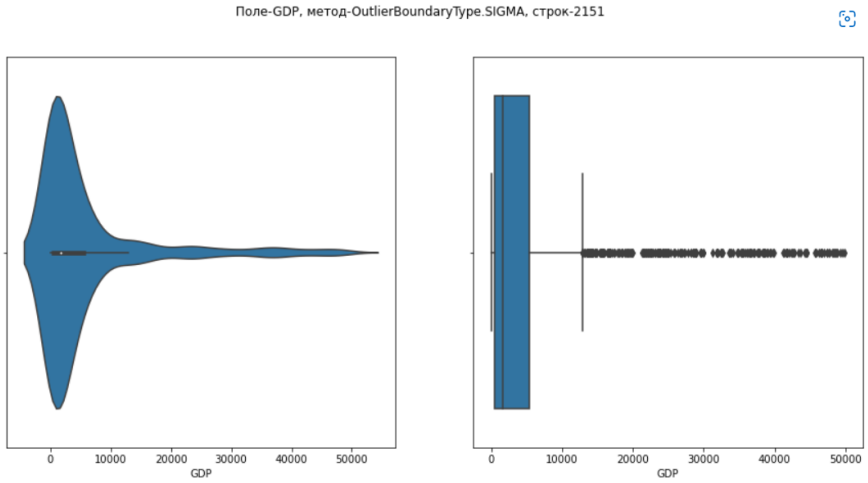
In [24]:
```
def plot_for_analys(df, variable, title):
    fig, ax = plt.subplots(figsize=(15,7))
    plt.subplot(1, 2, 1)
    sns.violinplot(x=df[variable])
    plt.subplot(1, 2, 2)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

In [25]:
```
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
def get_outlier_boundaries(df, col, outlier_boundary_type: OutlierBoundaryType):
    if outlier_boundary_type == OutlierBoundaryType.SIGMA:
        K1 = 3
        lower_boundary = df[col].mean() - (K1 * df[col].std())
        upper_boundary = df[col].mean() + (K1 * df[col].std())

    else:
        raise NameError('Unknown Outlier Boundary Type')

    return lower_boundary, upper_boundary
```

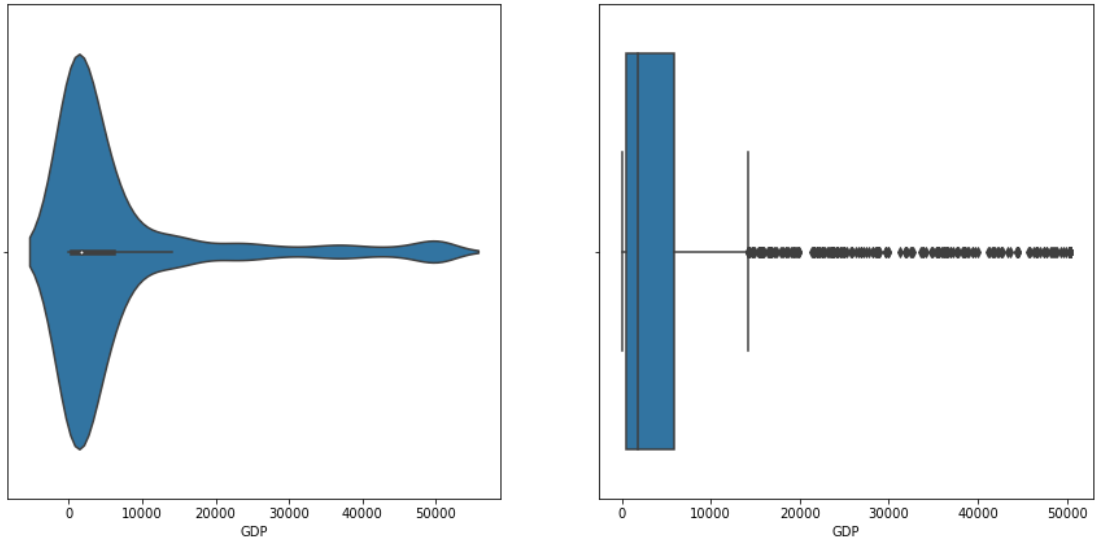
In [26]:
```
x_col_list = ['GDP']
data=X_train
for col in x_col_list:
    for obt in OutlierBoundaryType:
        lower_boundary, upper_boundary = get_outlier_boundaries(data, col, obt)
        # Флаги для удаления выбросов
        outliers_temp = np.where(data[col] > upper_boundary, True,
                                 np.where(data[col] < lower_boundary, True, False))
        # Удаление данных на основе флага
        data_trimmed = data.loc[~(outliers_temp), ]
        title = 'Поле-{}, метод-{}, строк-{}'.format(col, obt, data_trimmed.shape[0])
        plot_for_analys(data_trimmed, col, title)
```



3.2.2 Замена выбросов

In [27]:
```
for col in x_col_list:
    for obt in OutlierBoundaryType:
        lower_boundary, upper_boundary = get_outlier_boundaries(data, col, obt)
        data[col] = np.where(data[col] > upper_boundary, upper_boundary,
                             np.where(data[col] < lower_boundary, lower_boundary, data[col]))
        title = 'Поле-{}, метод-{}'.format(col, obt)
        plot_for_analys(data, col, title)
```

Поле-GDP, метод-OutlierBoundaryType.SIGMA



3.3 Обработка по крайней мере одного нестандартного признака

Преобразование текстовых данных в числовую матрицу

3.3.1 Word counts

In [30]:
```
dataset.head()
```

Out[30]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GD
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1	584.2592
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1	612.6965
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.7449
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	0.1	669.9590
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1	63.5372

5 rows × 22 columns

In [33]:
```
from sklearn.feature_extraction.text import CountVectorizer

data1=dataset['Country']
vec = CountVectorizer()
X = vec.fit_transform(data1)
df = pd.DataFrame(X.toarray(), columns=vec.get_feature_names())
df["text"] = data1
df
```

Out[33]:

	afghanistan	africa	african	albania	algeria	america	and	angola	antigua	arab	...	venezuela	verde	viet	vincent	yemen	yugoslav	zambia	zealand
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2933	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2934	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2935	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2936	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2937	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

2938 rows × 235 columns

3.3.2 TF-IDF(Term Frequency-Inverse Document Frequency)

In [34]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vec = TfidfVectorizer()
X = vec.fit_transform(data1)

df = pd.DataFrame(X.toarray(), columns=vec.get_feature_names())
df["text"] = data1
df
```

Out[34]:

	afghanistan	africa	african	albania	algeria	america	and	angola	antigua	arab	...	venezuela	verde	viet	vincent	yemen	yugoslav	zambia	zealand
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2933	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2934	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2935	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2936	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2937	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

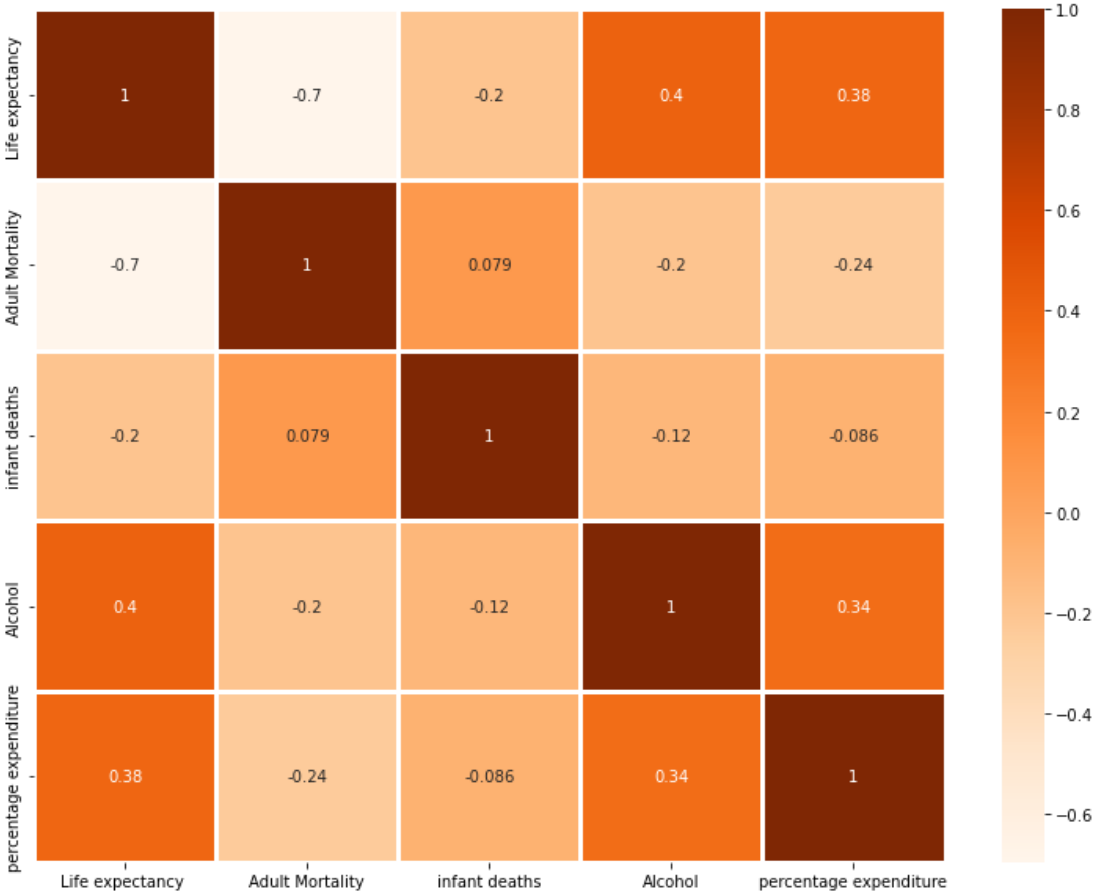
2938 rows × 235 columns

3.4 отбор признаков

3.4.1 один метод из группы методов фильтрации (filter methods);

In [51]:
```
data2=dataset.iloc[:,3:8]

plt.figure(figsize=(13,10))
sns.heatmap(data2.corr(), cmap="Oranges", annot=True, linewidths=3)
```



In [52]:
```
def make_corr_df(df):
    cr = data2.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= 0.53]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
    cr.columns = ['f1', 'f2', 'corr']
    return cr
```

In [53]:
```
make_corr_df(data2)
```

Out[53]:

	f1	f2	corr
0	Life expectancy	Adult Mortality	0.696359
1	Adult Mortality	Life expectancy	0.696359

In [54]:
```
def corr_groups(cr):
    grouped_feature_list = []
    correlated_groups = []

    for feature in cr['f1'].unique():
        if feature not in grouped_feature_list:
            # находим коррелирующие признаки
            correlated_block = cr[cr['f1'] == feature]
            cur_dups = list(correlated_block['f2'].unique()) + [feature]
            grouped_feature_list = grouped_feature_list + cur_dups
            correlated_groups.append(cur_dups)
    return correlated_groups
```

In [55]:
```
# Группы коррелирующих признаков
corr_groups(make_corr_df(data2))
```

Out[55]:
```
[['Adult Mortality', 'Life expectancy ']]
```

Эти два столбца имеют сильную корреляцию, удалите один из столбцов

In [56]:
```
data2=data2.drop(['Adult Mortality'],axis=1)
data2.head()
```

Out[56]:

	Life expectancy	infant deaths	Alcohol	percentage expenditure
0	65.0	62	0.01	71.279624
1	59.9	64	0.01	73.523582
2	59.9	66	0.01	73.219243
3	59.5	69	0.01	78.184215
4	59.2	71	0.01	7.097109

3.4.2 один метод из группы методов обертывания (wrapper methods)

заполнить пробелы

In [81]:
```
from sklearn.impute import SimpleImputer
imp_freq=SimpleImputer(strategy='most_frequent')
dataset['Adult Mortality']=imp_freq.fit_transform(dataset[['Adult Mortality']])
dataset['Alcohol']=imp_freq.fit_transform(dataset[['Alcohol']])
dataset['Polio']=imp_freq.fit_transform(dataset[['Polio']])
dataset['GDP']=imp_freq.fit_transform(dataset[['GDP']])
dataset['Population']=imp_freq.fit_transform(dataset[['Population']])
```

wrapper methods

In [85]:
```
from sklearn.neighbors import KNeighborsClassifier
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS


X=dataset[['Alcohol','Polio','GDP','Population']]
y=dataset[['Adult Mortality']]
X_train, X_test, y_train, y_test=train_test_split(X, y)
knn = KNeighborsClassifier(n_neighbors=3)
efs1 = EFS(knn,
           min_features=1,
           max_features=3,
           scoring='accuracy',
           print_progress=True,
           cv=5)

efs1 = efs1.fit(X_train, y_train, custom_feature_names=X.columns)

print('Best accuracy score: %.2f' % efs1.best_score_)
print('Best subset (indices):', efs1.best_idx_)
print('Best subset (corresponding names):', efs1.best_feature_names_)
```

```
Features: 14/14
```

```
Best accuracy score: 0.02
Best subset (indices): (0, 2, 3)
Best subset (corresponding names): ('Alcohol', 'GDP', 'Population')
```

3.4.3 один метод из группы методов вложений (embedded methods)

In [86]:
```
from sklearn.linear_model import Lasso
# Используем L1-регуляризацию
e_ls1 = Lasso(random_state=1)
e_ls1.fit(X_train, y_train)
# Коэффициенты регрессии
list(zip(X_train.columns, e_ls1.coef_))
```

Out[86]:
```
[('Alcohol', -2.4810128603610444),
 ('Polio', -1.1461474687966557),
 ('GDP', -0.0019963188916686447),
 ('Population', -2.4480409189060463e-08)]
```

In [87]:
```
from sklearn.feature_selection import SelectFromModel
sel_e_ls1 = SelectFromModel(e_ls1)
sel_e_ls1.fit(X_train, y_train)
list(zip(X_train.columns, sel_e_ls1.get_support()))
```

Out[87]:
```
[('Alcohol', True), ('Polio', True), ('GDP', True), ('Population', False)]
```