In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [2]:

```
path = 'C:/Users/79772/Desktop/3 курс/6 сем/ТМО/Лабы/Лаб. 2/20220327 annual-number-of-deaths-by-cause.csv'
data = pd.read_csv(path, sep=',')
```

In [3]:

```
data.shape
```

Out[3]:

```
(8254, 36)
```

In [4]:

```
data.dtypes
```

Out[4]:

```
Entity                                                                object
Code                                                                   object
Year                                                                    int64
Number of executions (Amnesty International)                           object
Deaths - Meningitis - Sex: Both - Age: All Ages (Number)               float64
Deaths - Neoplasms - Sex: Both - Age: All Ages (Number)                float64
Deaths - Fire, heat, and hot substances - Sex: Both - Age: All Ages (Number)    float64
Deaths - Malaria - Sex: Both - Age: All Ages (Number)                  float64
Deaths - Drowning - Sex: Both - Age: All Ages (Number)                 float64
Deaths - Interpersonal violence - Sex: Both - Age: All Ages (Number)    float64
Deaths - HIV/AIDS - Sex: Both - Age: All Ages (Number)                 float64
Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)       float64
Deaths - Tuberculosis - Sex: Both - Age: All Ages (Number)             float64
Deaths - Road injuries - Sex: Both - Age: All Ages (Number)            float64
Deaths - Maternal disorders - Sex: Both - Age: All Ages (Number)       float64
Deaths - Lower respiratory infections - Sex: Both - Age: All Ages (Number)    float64
Deaths - Neonatal disorders - Sex: Both - Age: All Ages (Number)       float64
Deaths - Alcohol use disorders - Sex: Both - Age: All Ages (Number)    float64
Deaths - Exposure to forces of nature - Sex: Both - Age: All Ages (Number)    float64
Deaths - Diarrheal diseases - Sex: Both - Age: All Ages (Number)       float64
Deaths - Environmental heat and cold exposure - Sex: Both - Age: All Ages (Number)    float64
Deaths - Nutritional deficiencies - Sex: Both - Age: All Ages (Number)    float64
Deaths - Self-harm - Sex: Both - Age: All Ages (Number)                float64
Deaths - Conflict and terrorism - Sex: Both - Age: All Ages (Number)    float64
Deaths - Diabetes mellitus - Sex: Both - Age: All Ages (Number)         float64
Deaths - Poisonings - Sex: Both - Age: All Ages (Number)                float64
Deaths - Protein-energy malnutrition - Sex: Both - Age: All Ages (Number)    float64
Terrorism (deaths)                                                     float64
Deaths - Cardiovascular diseases - Sex: Both - Age: All Ages (Number)   float64
Deaths - Chronic kidney disease - Sex: Both - Age: All Ages (Number)    float64
Deaths - Chronic respiratory diseases - Sex: Both - Age: All Ages (Number)    float64
Deaths - Cirrhosis and other chronic liver diseases - Sex: Both - Age: All Ages (Number)    float64
Deaths - Digestive diseases - Sex: Both - Age: All Ages (Number)       float64
Deaths - Acute hepatitis - Sex: Both - Age: All Ages (Number)          float64
Deaths - Alzheimer's disease and other dementias - Sex: Both - Age: All Ages (Number)    float64
Deaths - Parkinson's disease - Sex: Both - Age: All Ages (Number)       float64
dtype: object
```

In [5]:

```
data.isnull().sum()
```

Out[5]:

```
Entity                                                                  0
Code                                                                 2048
Year                                                                    0
Number of executions (Amnesty International)                         7987
Deaths - Meningitis - Sex: Both - Age: All Ages (Number)              244
Deaths - Neoplasms - Sex: Both - Age: All Ages (Number)               244
Deaths - Fire, heat, and hot substances - Sex: Both - Age: All Ages (Number)    244
Deaths - Malaria - Sex: Both - Age: All Ages (Number)                 244
Deaths - Drowning - Sex: Both - Age: All Ages (Number)                244
Deaths - Interpersonal violence - Sex: Both - Age: All Ages (Number)   244
Deaths - HIV/AIDS - Sex: Both - Age: All Ages (Number)                244
Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)      244
Deaths - Tuberculosis - Sex: Both - Age: All Ages (Number)            244
Deaths - Road injuries - Sex: Both - Age: All Ages (Number)           244
Deaths - Maternal disorders - Sex: Both - Age: All Ages (Number)      244
Deaths - Lower respiratory infections - Sex: Both - Age: All Ages (Number)   244
Deaths - Neonatal disorders - Sex: Both - Age: All Ages (Number)      244
Deaths - Alcohol use disorders - Sex: Both - Age: All Ages (Number)   244
Deaths - Exposure to forces of nature - Sex: Both - Age: All Ages (Number)   244
Deaths - Diarrheal diseases - Sex: Both - Age: All Ages (Number)      244
Deaths - Environmental heat and cold exposure - Sex: Both - Age: All Ages (Number)   244
Deaths - Nutritional deficiencies - Sex: Both - Age: All Ages (Number)   244
Deaths - Self-harm - Sex: Both - Age: All Ages (Number)               244
Deaths - Conflict and terrorism - Sex: Both - Age: All Ages (Number)  244
Deaths - Diabetes mellitus - Sex: Both - Age: All Ages (Number)       244
Deaths - Poisonings - Sex: Both - Age: All Ages (Number)              244
Deaths - Protein-energy malnutrition - Sex: Both - Age: All Ages (Number)   244
Terrorism (deaths)                                                   5363
Deaths - Cardiovascular diseases - Sex: Both - Age: All Ages (Number)  244
Deaths - Chronic kidney disease - Sex: Both - Age: All Ages (Number)   244
Deaths - Chronic respiratory diseases - Sex: Both - Age: All Ages (Number)   244
Deaths - Cirrhosis and other chronic liver diseases - Sex: Both - Age: All Ages (Number)   244
Deaths - Digestive diseases - Sex: Both - Age: All Ages (Number)      244
Deaths - Acute hepatitis - Sex: Both - Age: All Ages (Number)         244
Deaths - Alzheimer's disease and other dementias - Sex: Both - Age: All Ages (Number)   244
Deaths - Parkinson's disease - Sex: Both - Age: All Ages (Number)     244
dtype: int64
```

In [6]:

```
data.head()
```

Out[6]:

| | Entity | Code | Year | Number of executions (Amnesty International) | Deaths - Meningitis - Sex: Both - Age: All Ages (Number) | Deaths - Neoplasms - Sex: Both - Age: All Ages (Number) | Deaths - Fire, heat, and hot substances - Sex: Both - Age: All Ages (Number) | Deaths - Malaria - Sex: Both - Age: All Ages (Number) | Deaths - Drowning - Sex: Both - Age: All Ages (Number) | Deaths - Interpersonal violence - Sex: Both - Age: All Ages (Number) | ... | Deaths - Protein-energy malnutrition - Sex: Both - Age: All Ages (Number) | Terrorism (deaths) | Deaths - Cardiovascular diseases - Sex: Both - Age: All Ages (Number) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | AFG | 2007 | 15 | 2933.0 | 15925.0 | 481.0 | 393.0 | 2127.0 | 3657.0 | ... | 2439.0 | 1199.0 | 53962.0 |
| **1** | Afghanistan | AFG | 2008 | 17 | 2731.0 | 16148.0 | 462.0 | 255.0 | 1973.0 | 3785.0 | ... | 2231.0 | 1092.0 | 54051.0 |
| **2** | Afghanistan | AFG | 2009 | 0 | 2460.0 | 16383.0 | 448.0 | 239.0 | 1852.0 | 3874.0 | ... | 1998.0 | 1065.0 | 53964.0 |
| **3** | Afghanistan | AFG | 2011 | 2 | 2327.0 | 17094.0 | 448.0 | 390.0 | 1775.0 | 4170.0 | ... | 1805.0 | 1525.0 | 54347.0 |
| **4** | Afghanistan | AFG | 2012 | 14 | 2254.0 | 17522.0 | 445.0 | 94.0 | 1716.0 | 4245.0 | ... | 1667.0 | 3521.0 | 54868.0 |

5 rows × 36 columns

In [7]:

```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

Всего строк: 8254

# Обработка пропусков данных

# Удаление колонок с пустыми значениями

In [8]:

```
data_new_1 = data.dropna(axis=1, how='any')
```

```
(data.shape, data_new_1.shape)
```

Out[8]:

```
((8254, 36), (8254, 2))
```

# Удаление строк с пустыми значениями

In [9]:

```
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

Out[9]:

```
((8254, 36), (176, 36))
```

# Заполнение пропусков нулями

In [10]:

```
data_new_3 = data.fillna(0)
data_new_3.head()
```

Out[10]:

| | Entity | Code | Year | Number of executions (Amnesty International) | Deaths - Meningitis - Sex: Both - Age: All Ages (Number) | Deaths - Neoplasms - Sex: Both - Age: All Ages (Number) | Deaths - Fire, heat, and hot substances - Sex: Both - Age: All Ages (Number) | Deaths - Malaria - Sex: Both - Age: All Ages (Number) | Deaths - Drowning - Sex: Both - Age: All Ages (Number) | Deaths - Interpersonal violence - Sex: Both - Age: All Ages (Number) | ... | Deaths - Protein-energy malnutrition - Sex: Both - Age: All Ages (Number) | Terrorism (deaths) | Deaths - Cardiovascular diseases - Sex: Both - Age: All Ages (Number) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | AFG | 2007 | 15 | 2933.0 | 15925.0 | 481.0 | 393.0 | 2127.0 | 3657.0 | ... | 2439.0 | 1199.0 | 53962.0 |
| **1** | Afghanistan | AFG | 2008 | 17 | 2731.0 | 16148.0 | 462.0 | 255.0 | 1973.0 | 3785.0 | ... | 2231.0 | 1092.0 | 54051.0 |
| **2** | Afghanistan | AFG | 2009 | 0 | 2460.0 | 16383.0 | 448.0 | 239.0 | 1852.0 | 3874.0 | ... | 1998.0 | 1065.0 | 53964.0 |
| **3** | Afghanistan | AFG | 2011 | 2 | 2327.0 | 17094.0 | 448.0 | 390.0 | 1775.0 | 4170.0 | ... | 1805.0 | 1525.0 | 54347.0 |
| **4** | Afghanistan | AFG | 2012 | 14 | 2254.0 | 17522.0 | 445.0 | 94.0 | 1716.0 | 4245.0 | ... | 1667.0 | 3521.0 | 54868.0 |

5 rows × 36 columns

# "Внедрение значений" - импьютация (imputation)

Обработка пропусков в числовых данных

In [11]:

```
# Выберем числовые колонки 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 / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

Колонка Deaths - Meningitis - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Neoplasms - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Fire, heat, and hot substances - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Malaria - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Drowning - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Interpersonal violence - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - HIV/AIDS - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Tuberculosis - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Road injuries - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Maternal disorders - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Lower respiratory infections - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Neonatal disorders - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Alcohol use disorders - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Exposure to forces of nature - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Diarrheal diseases - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Environmental heat and cold exposure - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96
%.
Колонка Deaths - Nutritional deficiencies - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Self-harm - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Conflict and terrorism - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Diabetes mellitus - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Poisonings - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Protein-energy malnutrition - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Terrorism (deaths). Тип данных float64. Количество пустых значений 5363, 64.97%.
Колонка Deaths - Cardiovascular diseases - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Chronic kidney disease - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Chronic respiratory diseases - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Cirrhosis and other chronic liver diseases - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2
.96%.
Колонка Deaths - Digestive diseases - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Acute hepatitis - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.
Колонка Deaths - Alzheimer's disease and other dementias - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2
.96%.
Колонка Deaths - Parkinson's disease - Sex: Both - Age: All Ages (Number). Тип данных float64. Количество пустых значений 244, 2.96%.

In [12]:

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

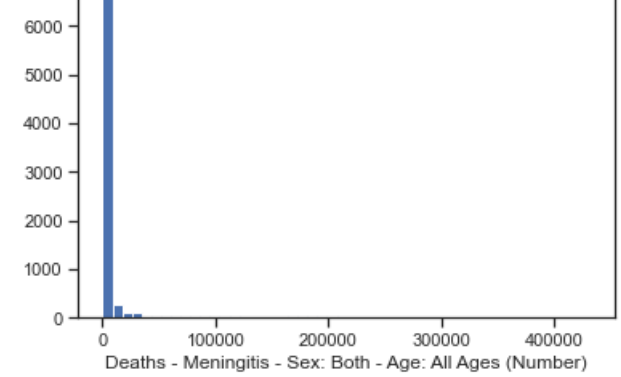
Out[12]:

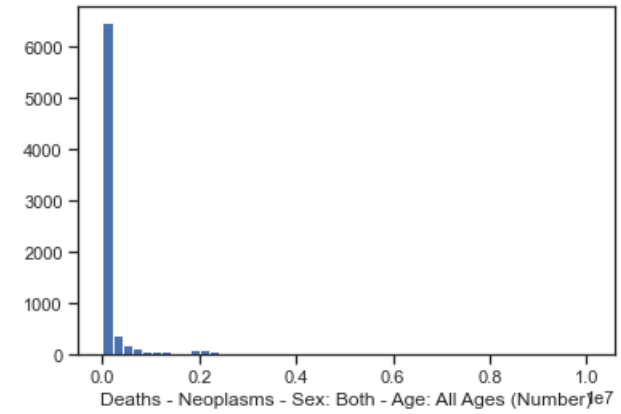
| | Deaths - Meningitis - Sex: Both - Age: All Ages (Number) | Deaths - Neoplasms - Sex: Both - Age: All Ages (Number) | Deaths - Fire, heat, and hot substances - Sex: Both - Age: All Ages (Number) | Deaths - Malaria - Sex: Both - Age: All Ages (Number) | Deaths - Drowning - Sex: Both - Age: All Ages (Number) | Deaths - Interpersonal violence - Sex: Both - Age: All Ages (Number) | Deaths - HIV/AIDS - Sex: Both - Age: All Ages (Number) | Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number) | Deaths - Tuberculosis - Sex: Both - Age: All Ages (Number) | Deaths - Road injuries - Sex: Both - Age: All Ages (Number) | ... | Deaths - Protein-energy malnutrition - Sex: Both - Age: All Ages (Number) | Terrorism (deaths) | Cardi disea Both Ages |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2933.0 | 15925.0 | 481.0 | 393.0 | 2127.0 | 3657.0 | 148.0 | 252.0 | 4995.0 | 7425.0 | ... | 2439.0 | 1199.0 | |
| **1** | 2731.0 | 16148.0 | 462.0 | 255.0 | 1973.0 | 3785.0 | 157.0 | 261.0 | 4790.0 | 7355.0 | ... | 2231.0 | 1092.0 | |
| **2** | 2460.0 | 16383.0 | 448.0 | 239.0 | 1852.0 | 3874.0 | 167.0 | 270.0 | 4579.0 | 7290.0 | ... | 1998.0 | 1065.0 | |
| **3** | 2327.0 | 17094.0 | 448.0 | 390.0 | 1775.0 | 4170.0 | 184.0 | 292.0 | 4259.0 | 7432.0 | ... | 1805.0 | 1525.0 | |
| **4** | 2254.0 | 17522.0 | 445.0 | 94.0 | 1716.0 | 4245.0 | 191.0 | 305.0 | 4122.0 | 7494.0 | ... | 1667.0 | 3521.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **8249** | 1439.0 | 11161.0 | 632.0 | 2518.0 | 770.0 | 1302.0 | 29162.0 | 104.0 | 11214.0 | 2373.0 | ... | 2990.0 | NaN | |
| **8250** | 1457.0 | 11465.0 | 648.0 | 2050.0 | 801.0 | 1342.0 | 27141.0 | 110.0 | 10998.0 | 2436.0 | ... | 3027.0 | NaN | |
| **8251** | 1460.0 | 11744.0 | 654.0 | 2116.0 | 818.0 | 1363.0 | 24846.0 | 115.0 | 10762.0 | 2473.0 | ... | 2962.0 | 0.0 | |
| **8252** | 1450.0 | 12038.0 | 657.0 | 2088.0 | 825.0 | 1396.0 | 22106.0 | 121.0 | 10545.0 | 2509.0 | ... | 2890.0 | NaN | |
| **8253** | 1450.0 | 12353.0 | 662.0 | 2068.0 | 827.0 | 1434.0 | 20722.0 | 127.0 | 10465.0 | 2554.0 | ... | 2855.0 | NaN | |

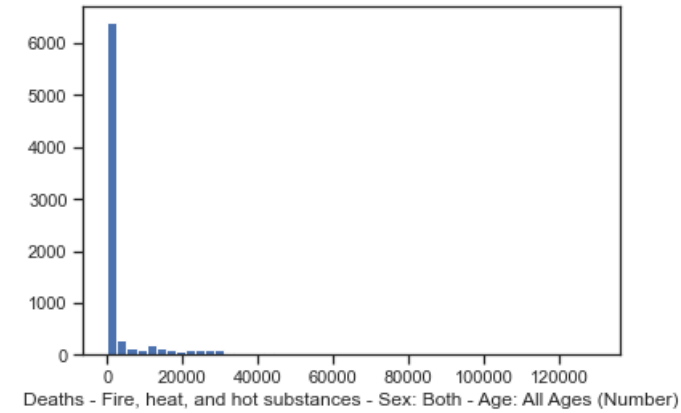
8254 rows × 32 columns

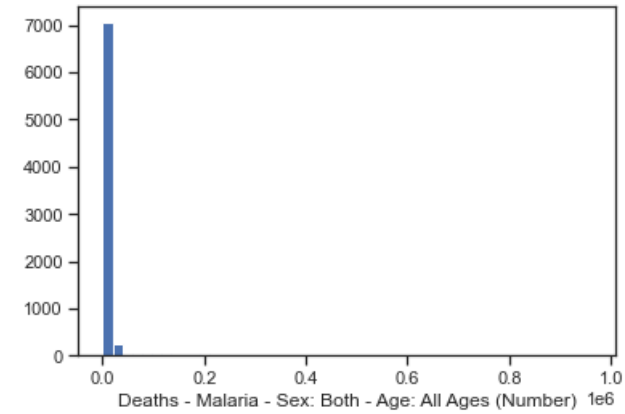
In [13]:

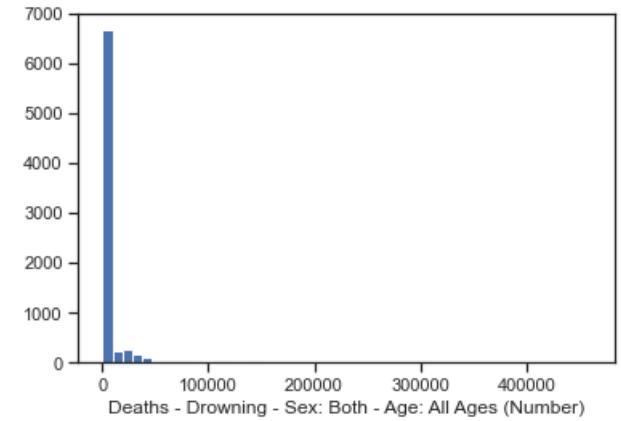
```
# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```


6000
5000
4000
3000
2000
1000
0
0
100000
200000
300000
400000
Deaths - Meningitis - Sex: Both - Age: All Ages (Number)

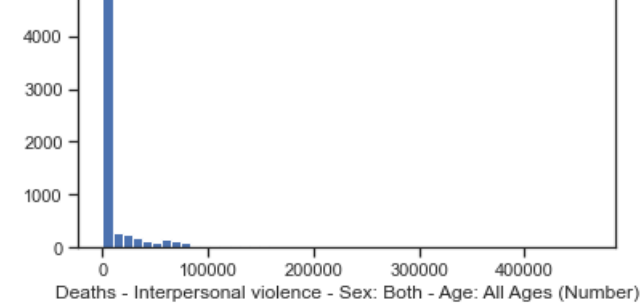

6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
Deaths - Neoplasms - Sex: Both - Age: All Ages (Number)
1e7

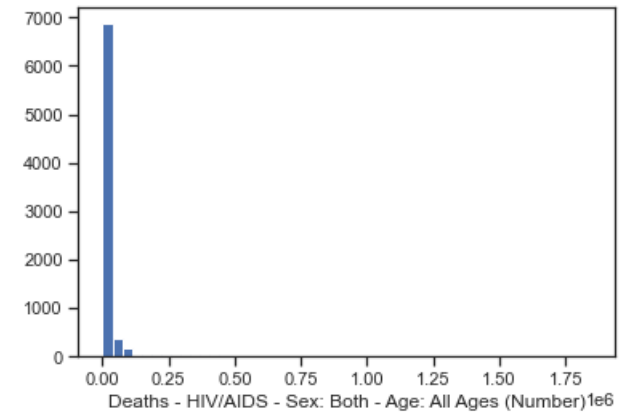

6000
5000
4000
3000
2000
1000
0
0
20000
40000
60000
80000
100000
120000
Deaths - Fire, heat, and hot substances - Sex: Both - Age: All Ages (Number)

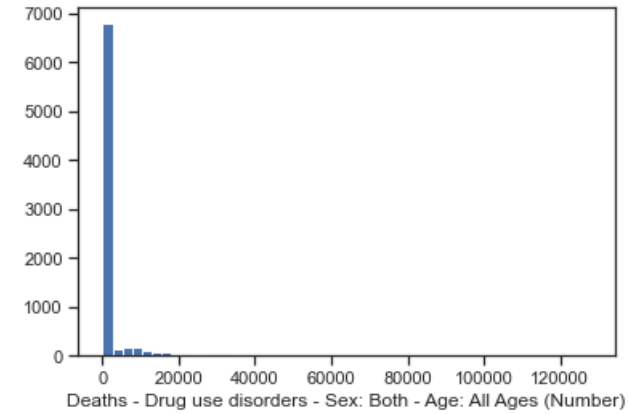

7000
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
Deaths - Malaria - Sex: Both - Age: All Ages (Number)
1e6

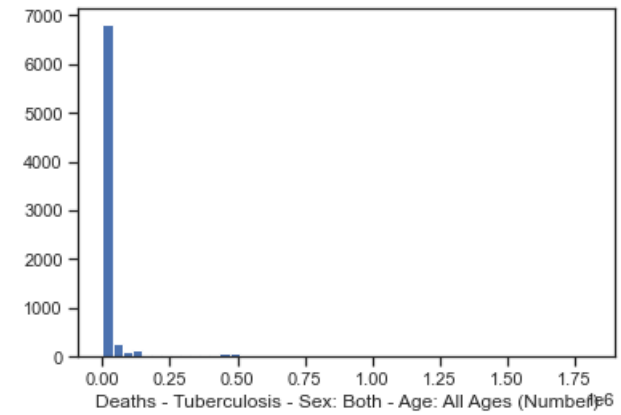

7000
6000
5000
4000
3000
2000
1000
0
0
100000
200000
300000
400000
Deaths - Drowning - Sex: Both - Age: All Ages (Number)

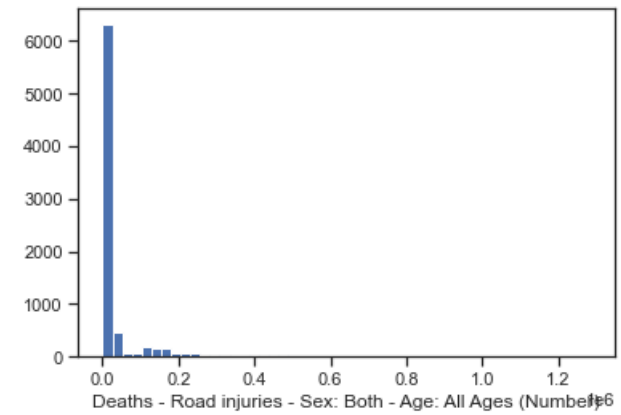

6000
5000


4000
3000
2000
1000
0
0
100000
200000
300000
400000
Deaths - Interpersonal violence - Sex: Both - Age: All Ages (Number)

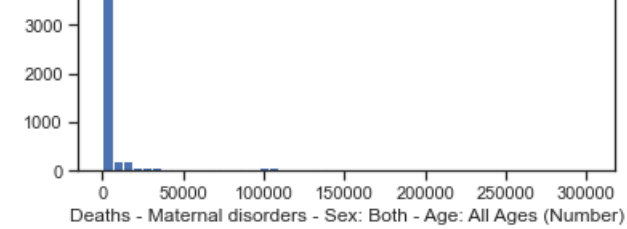

7000
6000
5000
4000
3000
2000
1000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Deaths - HIV/AIDS - Sex: Both - Age: All Ages (Number)
1e6


7000
6000
5000
4000
3000
2000
1000
0
0
20000
40000
60000
80000
100000
120000
Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)

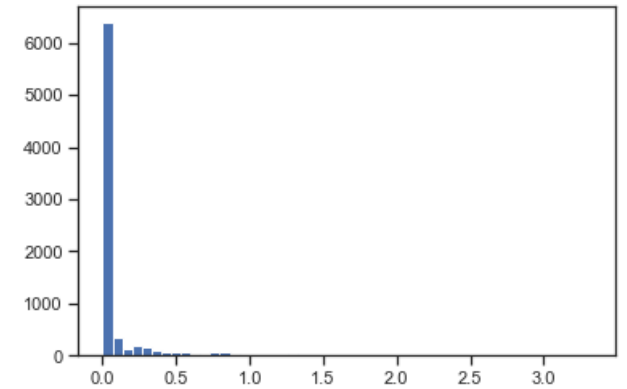

7000
6000
5000
4000
3000
2000
1000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Deaths - Tuberculosis - Sex: Both - Age: All Ages (Number)
1e6


6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Deaths - Road injuries - Sex: Both - Age: All Ages (Number)
1e6

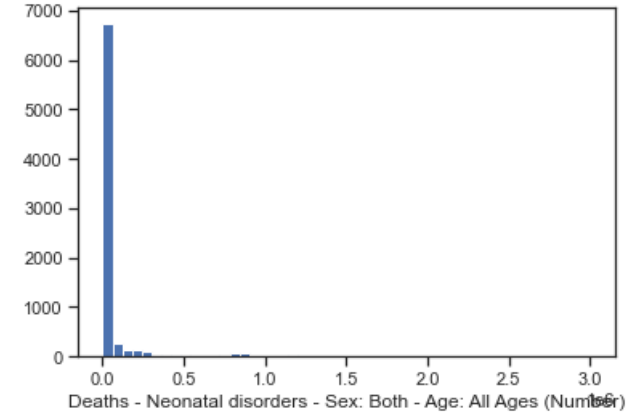

7000
6000
5000
4000



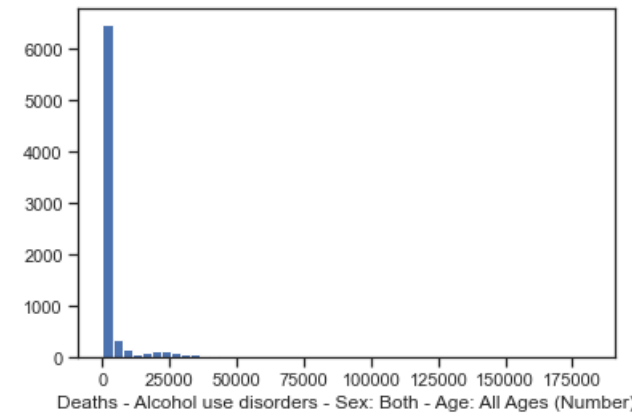
Deaths - Maternal disorders - Sex: Both - Age: All Ages (Number)



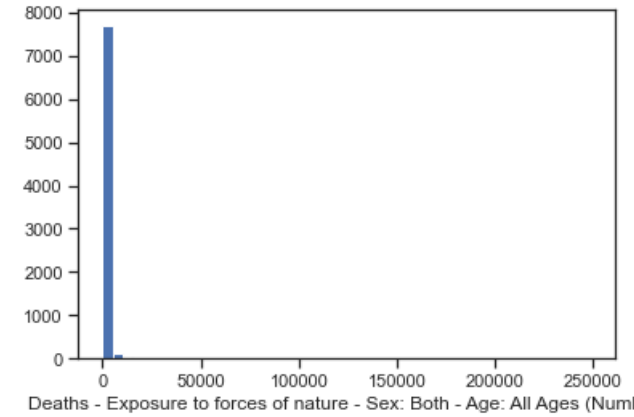
Deaths - Lower respiratory infections - Sex: Both - Age: All Ages (Number)



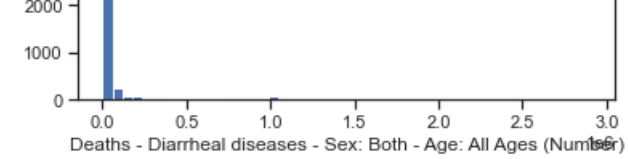
Deaths - Neonatal disorders - Sex: Both - Age: All Ages (Number)



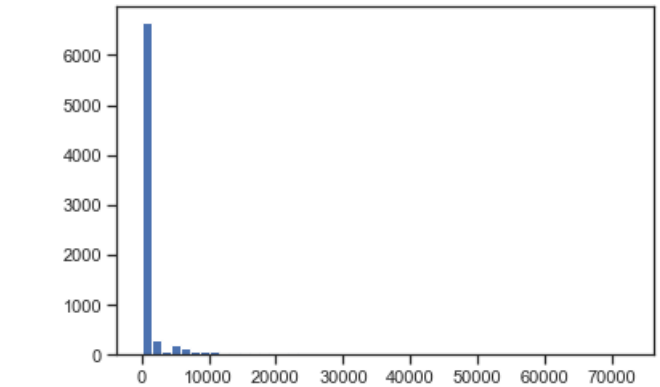
Deaths - Alcohol use disorders - Sex: Both - Age: All Ages (Number)



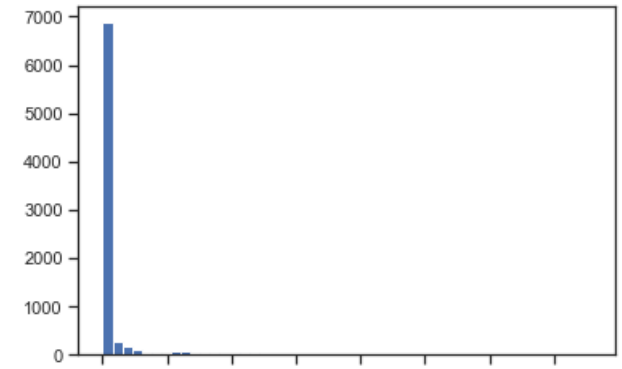
Deaths - Exposure to forces of nature - Sex: Both - Age: All Ages (Number)



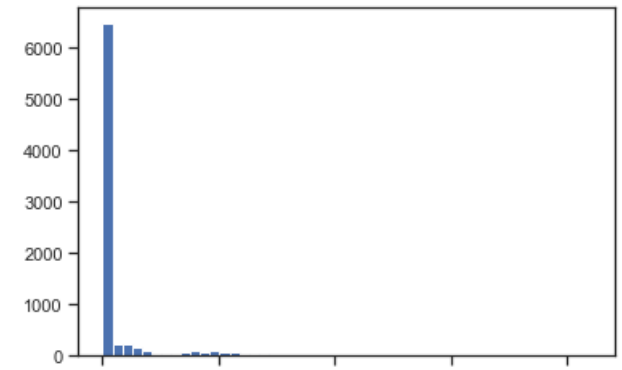
Deaths - Diarrheal diseases - Sex: Both - Age: All Ages (Number)



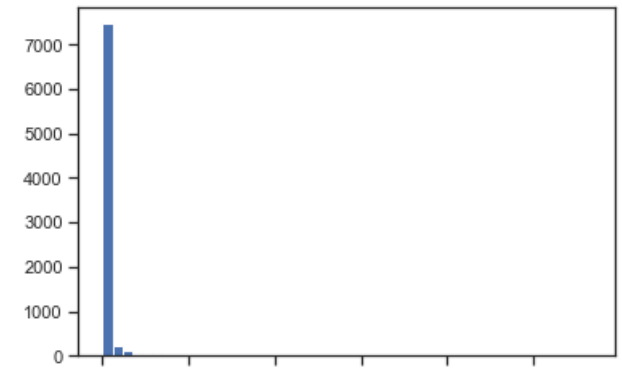
Deaths - Environmental heat and cold exposure - Sex: Both - Age: All Ages (Number)



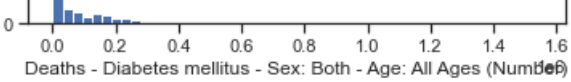
Deaths - Nutritional deficiencies - Sex: Both - Age: All Ages (Number)

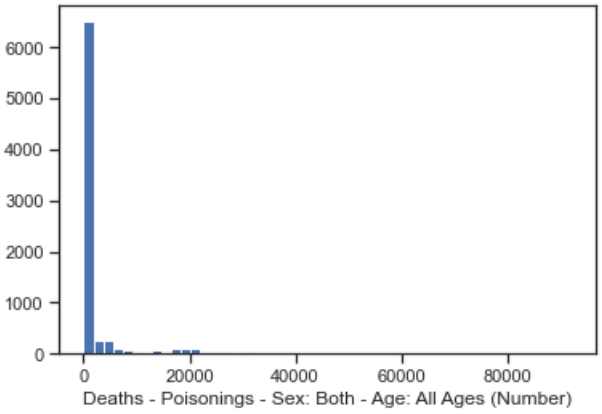


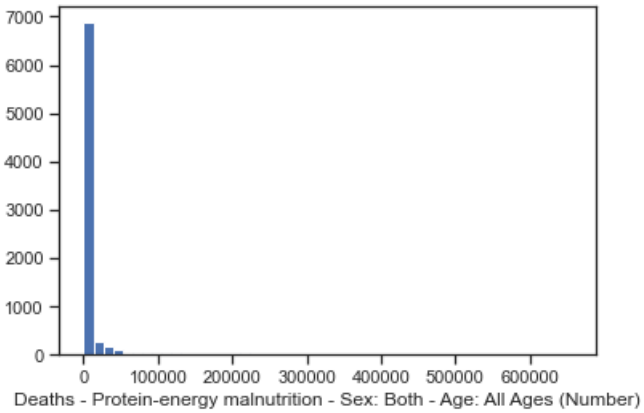
Deaths - Self-harm - Sex: Both - Age: All Ages (Number)

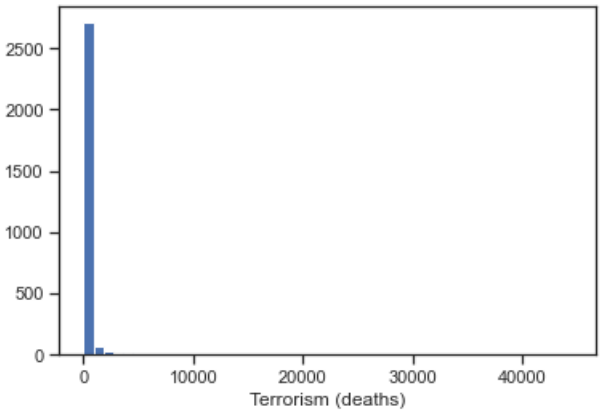


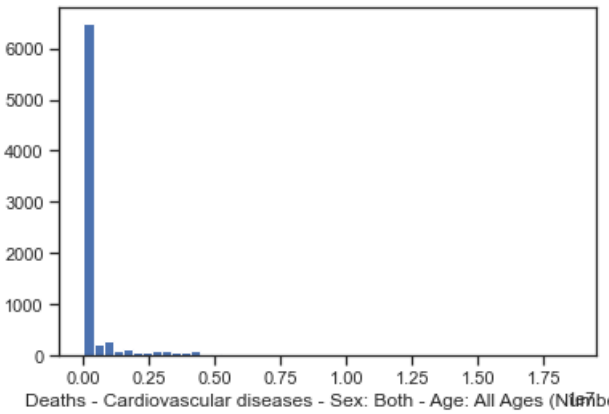
Deaths - Conflict and terrorism - Sex: Both - Age: All Ages (Number)

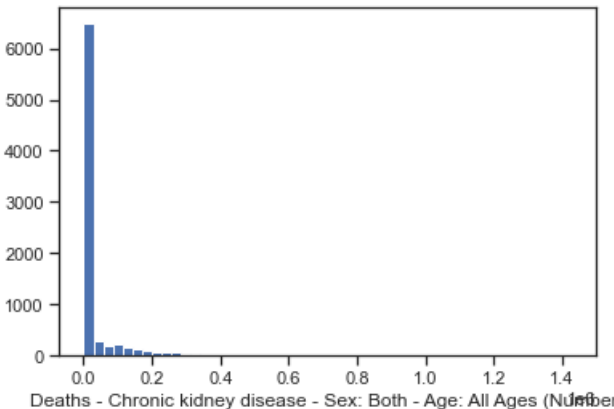

0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1e6
Deaths - Diabetes mellitus - Sex: Both - Age: All Ages (Number)

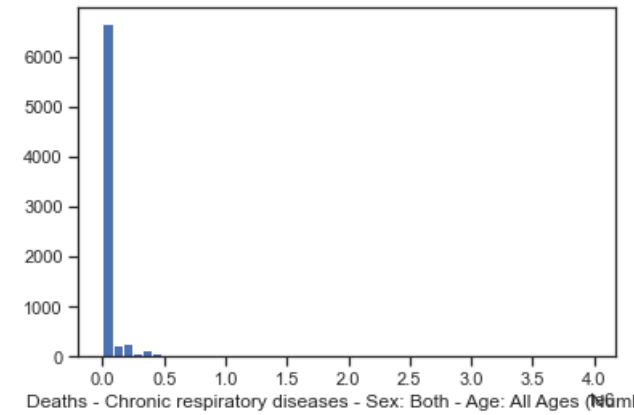

6000
5000
4000
3000
2000
1000
0
0
20000
40000
60000
80000
Deaths - Poisonings - Sex: Both - Age: All Ages (Number)


7000
6000
5000
4000
3000
2000
1000
0
0
100000
200000
300000
400000
500000
600000
Deaths - Protein-energy malnutrition - Sex: Both - Age: All Ages (Number)

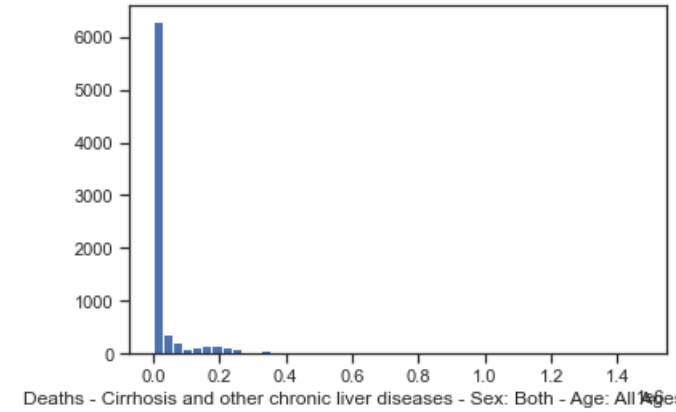

2500
2000
1500
1000
500
0
0
10000
20000
30000
40000
Terrorism (deaths)


6000
5000
4000
3000
2000
1000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e7
Deaths - Cardiovascular diseases - Sex: Both - Age: All Ages (Number)

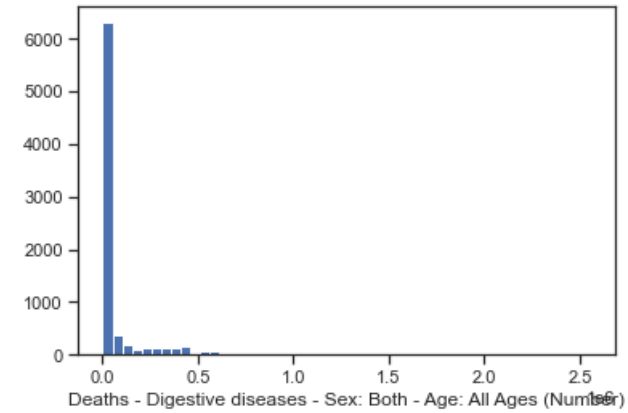

6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6
Deaths - Chronic kidney disease - Sex: Both - Age: All Ages (Number)



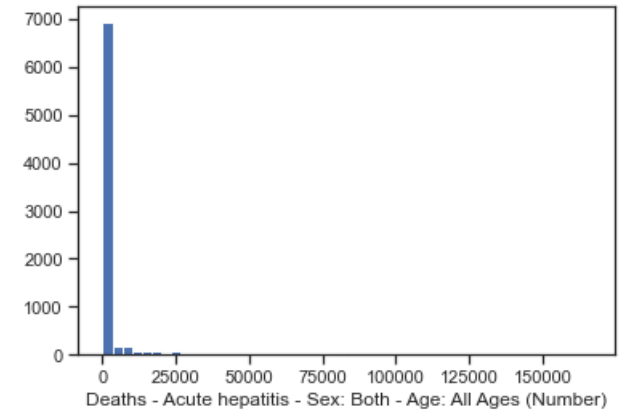
Deaths - Chronic respiratory diseases - Sex: Both - Age: All Ages (Number)



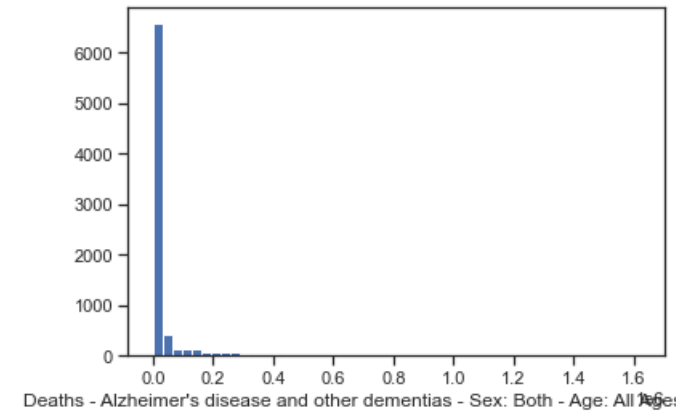
Deaths - Cirrhosis and other chronic liver diseases - Sex: Both - Age: All Ages (Number)



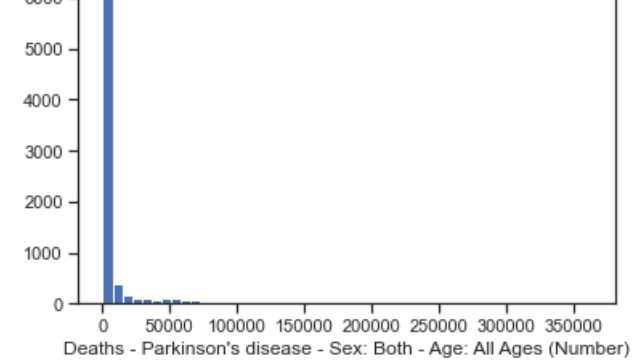
Deaths - Digestive diseases - Sex: Both - Age: All Ages (Number)



Deaths - Acute hepatitis - Sex: Both - Age: All Ages (Number)



Deaths - Alzheimer's disease and other dementias - Sex: Both - Age: All Ages (Number)




Будем использовать встроенные средства импьютации библиотеки scikit-learn - https://scikit-learn.org/stable/modules/impute.html

In [14]:

```
data_num_Terror = data_num[['Terrorism (deaths)']]
data_num_Terror.head()
```

Out[14]:

| | Terrorism (deaths) |
|---|---|
| **0** | 1199.0 |
| **1** | 1092.0 |
| **2** | 1065.0 |
| **3** | 1525.0 |
| **4** | 3521.0 |

In [15]:

```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

In [16]:

```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_Terror)
mask_missing_values_only
```

Out[16]:

```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [ True],
       [ True]])
```

С помощью класса SimpleImputer можно проводить импьютацию различными показателями центра распределения

In [17]:

```
strategies=['mean', 'median', 'most_frequent']
```

In [18]:

```
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(data_num_Terror)
    return data_num_imp[mask_missing_values_only]
```

In [19]:

```
strategies[0], test_num_impute(strategies[0])
```

Out[19]:

```
('mean',
 array([349.23590453, 349.23590453, 349.23590453, ..., 349.23590453,
        349.23590453, 349.23590453]))
```

In [20]:

```
strategies[1], test_num_impute(strategies[1])
```

Out[20]:

```
('median', array([5., 5., 5., ..., 5., 5., 5.]))
```

In [21]:

```
strategies[2], test_num_impute(strategies[2])
```

Out[21]:

```
('most_frequent', array([0., 0., 0., ..., 0., 0., 0.]))
```

In [22]:

```
# Более сложная функция, которая позволяет задавать колонку и вид импьютации
def test_num_impute_col(dataset, column, strategy_param):
    temp_data = dataset[[column]]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(temp_data)

    filled_data = data_num_imp[mask_missing_values_only]

    return column, strategy_param, filled_data.size, filled_data[0], filled_data[filled_data.size-1]
```

In [23]:

```
data[['Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)']].describe()
```

Out[23]:

| | Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number) |
|---|---|
| **count** | 8010.000000 |
| **mean** | 3469.958926 |
| **std** | 11186.514866 |
| **min** | 0.000000 |
| **25%** | 7.000000 |
| **50%** | 57.000000 |
| **75%** | 518.750000 |
| **max** | 128083.000000 |

In [24]:

```
test_num_impute_col(data, 'Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)', strategies[0])
```

Out[24]:

```
('Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)',
 'mean',
 244,
 3469.9589263420726,
 3469.9589263420726)
```

In [25]:

```
test_num_impute_col(data, 'Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)', strategies[1])
```

Out[25]:

```
('Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)',
 'median',
 244,
 57.0,
 57.0)
```

In [26]:

```
test_num_impute_col(data, 'Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)', strategies[2])
```

Out[26]:

```
('Deaths - Drug use disorders - Sex: Both - Age: All Ages (Number)',
 'most_frequent',
 244,
 0.0,
 0.0)
```

# Обработка пропусков в категориальных данных

In [27]:

```
# Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета
cat_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=='object'):
```

```
        cat_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

```
Колонка Code. Тип данных object. Количество пустых значений 2048, 24.81%.
Колонка Number of executions (Amnesty International). Тип данных object. Количество пустых значений 7987, 96.77%.
```

Класс SimpleImputer можно использовать для категориальных признаков со стратегиями "most_frequent" или "constant".

In [28]:

```
cat_temp_data = data[['Code']]
cat_temp_data.head()
```

Out[28]:

| | Code |
|---|---|
| **0** | AFG |
| **1** | AFG |
| **2** | AFG |
| **3** | AFG |
| **4** | AFG |

In [29]:

```
cat_temp_data['Code'].unique()
```

Out[29]:

```
array(['AFG', nan, 'ALB', 'DZA', 'ASM', 'AND', 'AGO', 'ATG', 'ARG', 'ARM',
       'AUS', 'AUT', 'AZE', 'BHS', 'BHR', 'BGD', 'BRB', 'BLR', 'BEL',
       'BLZ', 'BEN', 'BMU', 'BTN', 'BOL', 'BIH', 'BWA', 'BRA', 'BRN',
       'BGR', 'BFA', 'BDI', 'KHM', 'CMR', 'CAN', 'CPV', 'CAF', 'TCD',
       'CHL', 'CHN', 'COL', 'COM', 'COG', 'COK', 'CRI', 'CIV', 'HRV',
       'CUB', 'CYP', 'CZE', 'OWID_CZS', 'COD', 'DNK', 'DJI', 'DMA', 'DOM',
       'ECU', 'EGY', 'SLV', 'GNQ', 'ERI', 'EST', 'SWZ', 'ETH', 'FJI',
       'FIN', 'FRA', 'GUF', 'PYF', 'GAB', 'GMB', 'GEO', 'DEU', 'GHA',
       'GRC', 'GRL', 'GRD', 'GLP', 'GUM', 'GTM', 'GIN', 'GNB', 'GUY',
       'HTI', 'HND', 'HKG', 'HUN', 'ISL', 'IND', 'IDN', 'IRN', 'IRQ',
       'IRL', 'ISR', 'ITA', 'JAM', 'JPN', 'JOR', 'KAZ', 'KEN', 'KIR',
       'OWID_KOS', 'KWT', 'KGZ', 'LAO', 'LVA', 'LBN', 'LSO', 'LBR', 'LBY',
       'LTU', 'LUX', 'MDG', 'MWI', 'MYS', 'MDV', 'MLI', 'MLT', 'MHL',
       'MTQ', 'MRT', 'MUS', 'MEX', 'FSM', 'MDA', 'MCO', 'MNG', 'MNE',
       'MAR', 'MOZ', 'MMR', 'NAM', 'NRU', 'NPL', 'NLD', 'NCL', 'NZL',
       'NIC', 'NER', 'NGA', 'NIU', 'PRK', 'MKD', 'MNP', 'NOR', 'OMN',
       'PAK', 'PLW', 'PSE', 'PAN', 'PNG', 'PRY', 'PER', 'PHL', 'POL',
       'PRT', 'PRI', 'QAT', 'ROU', 'RUS', 'RWA', 'KNA', 'LCA', 'VCT',
       'WSM', 'SMR', 'STP', 'SAU', 'SEN', 'SRB', 'SYC', 'SLE', 'SGP',
       'SVK', 'SVN', 'SLB', 'SOM', 'ZAF', 'KOR', 'SSD', 'ESP', 'LKA',
       'SDN', 'SUR', 'SWE', 'CHE', 'SYR', 'TWN', 'TJK', 'TZA', 'THA',
       'TLS', 'TGO', 'TKL', 'TON', 'TTO', 'TUN', 'TUR', 'TKM', 'TUV',
       'OWID_USS', 'UGA', 'UKR', 'ARE', 'GBR', 'USA', 'VIR', 'URY', 'UZB',
       'VUT', 'VEN', 'VNM', 'WLF', 'ESH', 'OWID_WRL', 'YEM', 'OWID_YGS',
       'ZMB', 'ZWE'], dtype=object)
```

In [30]:

```
cat_temp_data[cat_temp_data['Code'].isnull()].shape
```

Out[30]:

```
(2048, 1)
```

In [31]:

```
# Импьютация наиболее частыми значениями
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

Out[31]:

```
array([['AFG'],
       ['AFG'],
       ['AFG'],
       ...,
       ['ZWE'],
       ['ZWE'],
       ['ZWE']], dtype=object)
```

In [32]:

```
# Пустые значения отсутствуют
np.unique(data_imp2)
```

Out[32]:

```
array(['AFG', 'AGO', 'ALB', 'AND', 'ARE', 'ARG', 'ARM', 'ASM', 'ATG',
       'AUS', 'AUT', 'AZE', 'BDI', 'BEL', 'BEN', 'BFA', 'BGD', 'BGR',
       'BHR', 'BHS', 'BIH', 'BLR', 'BLZ', 'BMU', 'BOL', 'BRA', 'BRB',
       'BRN', 'BTN', 'BWA', 'CAF', 'CAN', 'CHE', 'CHL', 'CHN', 'CIV',
       'CMR', 'COD', 'COG', 'COK', 'COL', 'COM', 'CPV', 'CRI', 'CUB',
       'CYP', 'CZE', 'DEU', 'DJI', 'DMA', 'DNK', 'DOM', 'DZA', 'ECU',
       'EGY', 'ERI', 'ESH', 'ESP', 'EST', 'ETH', 'FIN', 'FJI', 'FRA',
       'FSM', 'GAB', 'GBR', 'GEO', 'GHA', 'GIN', 'GLP', 'GMB', 'GNB',
       'GNQ', 'GRC', 'GRD', 'GRL', 'GTM', 'GUF', 'GUM', 'GUY', 'HKG',
       'HND', 'HRV', 'HTI', 'HUN', 'IDN', 'IND', 'IRL', 'IRN', 'IRQ',
       'ISL', 'ISR', 'ITA', 'JAM', 'JOR', 'JPN', 'KAZ', 'KEN', 'KGZ',
       'KHM', 'KIR', 'KNA', 'KOR', 'KWT', 'LAO', 'LBN', 'LBR', 'LBY',
       'LCA', 'LKA', 'LSO', 'LTU', 'LUX', 'LVA', 'MAR', 'MCO', 'MDA',
       'MDG', 'MDV', 'MEX', 'MHL', 'MKD', 'MLI', 'MLT', 'MMR', 'MNE',
       'MNG', 'MNP', 'MOZ', 'MRT', 'MTQ', 'MUS', 'MWI', 'MYS', 'NAM',
       'NCL', 'NER', 'NGA', 'NIC', 'NIU', 'NLD', 'NOR', 'NPL', 'NRU',
       'NZL', 'OMN', 'OWID_CZS', 'OWID_KOS', 'OWID_USS', 'OWID_WRL',
       'OWID_YGS', 'PAK', 'PAN', 'PER', 'PHL', 'PLW', 'PNG', 'POL', 'PRI',
       'PRK', 'PRT', 'PRY', 'PSE', 'PYF', 'QAT', 'ROU', 'RUS', 'RWA',
       'SAU', 'SDN', 'SEN', 'SGP', 'SLB', 'SLE', 'SLV', 'SMR', 'SOM',
       'SRB', 'SSD', 'STP', 'SUR', 'SVK', 'SVN', 'SWE', 'SWZ', 'SYC',
       'SYR', 'TCD', 'TGO', 'THA', 'TJK', 'TKL', 'TKM', 'TLS', 'TON',
       'TTO', 'TUN', 'TUR', 'TUV', 'TWN', 'TZA', 'UGA', 'UKR', 'URY',
       'USA', 'UZB', 'VCT', 'VEN', 'VIR', 'VNM', 'VUT', 'WLF', 'WSM',
       'YEM', 'ZAF', 'ZMB', 'ZWE'], dtype=object)
```

In [33]:

```
# Импьютация константой
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='NA')
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

Out[33]:

```
array([['AFG'],
       ['AFG'],
       ['AFG'],
       ...,
       ['ZWE'],
       ['ZWE'],
       ['ZWE']], dtype=object)
```

In [34]:

```
np.unique(data_imp3)
```

Out[34]:

```
array(['AFG', 'AGO', 'ALB', 'AND', 'ARE', 'ARG', 'ARM', 'ASM', 'ATG',
       'AUS', 'AUT', 'AZE', 'BDI', 'BEL', 'BEN', 'BFA', 'BGD', 'BGR',
       'BHR', 'BHS', 'BIH', 'BLR', 'BLZ', 'BMU', 'BOL', 'BRA', 'BRB',
       'BRN', 'BTN', 'BWA', 'CAF', 'CAN', 'CHE', 'CHL', 'CHN', 'CIV',
       'CMR', 'COD', 'COG', 'COK', 'COL', 'COM', 'CPV', 'CRI', 'CUB',
       'CYP', 'CZE', 'DEU', 'DJI', 'DMA', 'DNK', 'DOM', 'DZA', 'ECU',
       'EGY', 'ERI', 'ESH', 'ESP', 'EST', 'ETH', 'FIN', 'FJI', 'FRA',
       'FSM', 'GAB', 'GBR', 'GEO', 'GHA', 'GIN', 'GLP', 'GMB', 'GNB',
       'GNQ', 'GRC', 'GRD', 'GRL', 'GTM', 'GUF', 'GUM', 'GUY', 'HKG',
       'HND', 'HRV', 'HTI', 'HUN', 'IDN', 'IND', 'IRL', 'IRN', 'IRQ',
       'ISL', 'ISR', 'ITA', 'JAM', 'JOR', 'JPN', 'KAZ', 'KEN', 'KGZ',
       'KHM', 'KIR', 'KNA', 'KOR', 'KWT', 'LAO', 'LBN', 'LBR', 'LBY',
       'LCA', 'LKA', 'LSO', 'LTU', 'LUX', 'LVA', 'MAR', 'MCO', 'MDA',
       'MDG', 'MDV', 'MEX', 'MHL', 'MKD', 'MLI', 'MLT', 'MMR', 'MNE',
       'MNG', 'MNP', 'MOZ', 'MRT', 'MTQ', 'MUS', 'MWI', 'MYS', 'NA',
       'NAM', 'NCL', 'NER', 'NGA', 'NIC', 'NIU', 'NLD', 'NOR', 'NPL',
       'NRU', 'NZL', 'OMN', 'OWID_CZS', 'OWID_KOS', 'OWID_USS',
       'OWID_WRL', 'OWID_YGS', 'PAK', 'PAN', 'PER', 'PHL', 'PLW', 'PNG',
       'POL', 'PRI', 'PRK', 'PRT', 'PRY', 'PSE', 'PYF', 'QAT', 'ROU',
       'RUS', 'RWA', 'SAU', 'SDN', 'SEN', 'SGP', 'SLB', 'SLE', 'SLV',
       'SMR', 'SOM', 'SRB', 'SSD', 'STP', 'SUR', 'SVK', 'SVN', 'SWE',
       'SWZ', 'SYC', 'SYR', 'TCD', 'TGO', 'THA', 'TJK', 'TKL', 'TKM',
       'TLS', 'TON', 'TTO', 'TUN', 'TUR', 'TUV', 'TWN', 'TZA', 'UGA',
       'UKR', 'URY', 'USA', 'UZB', 'VCT', 'VEN', 'VIR', 'VNM', 'VUT',
       'WLF', 'WSM', 'YEM', 'ZAF', 'ZMB', 'ZWE'], dtype=object)
```

In [35]:

```
data_imp3[data_imp3=='NA'].size
```

Out[35]:

```
2048
```

# Преобразование категориальных признаков в числовые

In [36]:

```
cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
cat_enc
```

Out[36]:

| | c1 |
|---|---|
| **0** | AFG |
| **1** | AFG |
| **2** | AFG |
| **3** | AFG |
| **4** | AFG |
| **...** | ... |
| **8249** | ZWE |
| **8250** | ZWE |
| **8251** | ZWE |
| **8252** | ZWE |
| **8253** | ZWE |

8254 rows × 1 columns

# Кодирование категорий целочисленными значениями (label encoding)

В этом случае уникальные значения категориального признака кодируются целыми числами.

В scikit-learn для такого кодирования используется два класса :

LabelEncoder - который ориентирован на применение к одному признаку. Этот класс прежде всего предназначен для кодирования целевого признака, но может быть также использован для последовательного кодирования отдельных нецелевых признаков. OrdinalEncoder - который ориентирован на применение к матрице объект-признак, то есть для кодирования матрицы нецелевых признаков.

# Использование LabelEncoder

In [37]:

```
from sklearn.preprocessing import LabelEncoder
```

In [38]:

```
cat_enc['c1'].unique()
```

Out[38]:

```
array(['AFG', 'ALB', 'DZA', 'ASM', 'AND', 'AGO', 'ATG', 'ARG', 'ARM',
       'AUS', 'AUT', 'AZE', 'BHS', 'BHR', 'BGD', 'BRB', 'BLR', 'BEL',
       'BLZ', 'BEN', 'BMU', 'BTN', 'BOL', 'BIH', 'BWA', 'BRA', 'BRN',
       'BGR', 'BFA', 'BDI', 'KHM', 'CMR', 'CAN', 'CPV', 'CAF', 'TCD',
       'CHL', 'CHN', 'COL', 'COM', 'COG', 'COK', 'CRI', 'CIV', 'HRV',
       'CUB', 'CYP', 'CZE', 'OWID_CZS', 'COD', 'DNK', 'DJI', 'DMA', 'DOM',
       'ECU', 'EGY', 'SLV', 'GNQ', 'ERI', 'EST', 'SWZ', 'ETH', 'FJI',
       'FIN', 'FRA', 'GUF', 'PYF', 'GAB', 'GMB', 'GEO', 'DEU', 'GHA',
       'GRC', 'GRL', 'GRD', 'GLP', 'GUM', 'GTM', 'GIN', 'GNB', 'GUY',
       'HTI', 'HND', 'HKG', 'HUN', 'ISL', 'IND', 'IDN', 'IRN', 'IRQ',
       'IRL', 'ISR', 'ITA', 'JAM', 'JPN', 'JOR', 'KAZ', 'KEN', 'KIR',
       'OWID_KOS', 'KWT', 'KGZ', 'LAO', 'LVA', 'LBN', 'LSO', 'LBR', 'LBY',
       'LTU', 'LUX', 'MDG', 'MWI', 'MYS', 'MDV', 'MLI', 'MLT', 'MHL',
       'MTQ', 'MRT', 'MUS', 'MEX', 'FSM', 'MDA', 'MCO', 'MNG', 'MNE',
       'MAR', 'MOZ', 'MMR', 'NAM', 'NRU', 'NPL', 'NLD', 'NCL', 'NZL',
       'NIC', 'NER', 'NGA', 'NIU', 'PRK', 'MKD', 'MNP', 'NOR', 'OMN',
       'PAK', 'PLW', 'PSE', 'PAN', 'PNG', 'PRY', 'PER', 'PHL', 'POL',
       'PRT', 'PRI', 'QAT', 'ROU', 'RUS', 'RWA', 'KNA', 'LCA', 'VCT',
       'WSM', 'SMR', 'STP', 'SAU', 'SEN', 'SRB', 'SYC', 'SLE', 'SGP',
       'SVK', 'SVN', 'SLB', 'SOM', 'ZAF', 'KOR', 'SSD', 'ESP', 'LKA',
       'SDN', 'SUR', 'SWE', 'CHE', 'SYR', 'TWN', 'TJK', 'TZA', 'THA',
       'TLS', 'TGO', 'TKL', 'TON', 'TTO', 'TUN', 'TUR', 'TKM', 'TUV',
       'OWID_USS', 'UGA', 'UKR', 'ARE', 'GBR', 'USA', 'VIR', 'URY', 'UZB',
       'VUT', 'VEN', 'VNM', 'WLF', 'ESH', 'OWID_WRL', 'YEM', 'OWID_YGS',
       'ZMB', 'ZWE'], dtype=object)
```

In [39]:

```
le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
```

In [40]:

```
# Наименования категорий в соответствии с порядковыми номерами

# Свойство называется classes, потому что предполагается что мы решаем
# задачу классификации и каждое значение категории соответствует
# какому-либо классу целевого признака

le.classes_
```

Out[40]:

```
array(['AFG', 'AGO', 'ALB', 'AND', 'ARE', 'ARG', 'ARM', 'ASM', 'ATG',
       'AUS', 'AUT', 'AZE', 'BDI', 'BEL', 'BEN', 'BFA', 'BGD', 'BGR',
       'BHR', 'BHS', 'BIH', 'BLR', 'BLZ', 'BMU', 'BOL', 'BRA', 'BRB',
       'BRN', 'BTN', 'BWA', 'CAF', 'CAN', 'CHE', 'CHL', 'CHN', 'CIV',
       'CMR', 'COD', 'COG', 'COK', 'COL', 'COM', 'CPV', 'CRI', 'CUB',
       'CYP', 'CZE', 'DEU', 'DJI', 'DMA', 'DNK', 'DOM', 'DZA', 'ECU',
       'EGY', 'ERI', 'ESH', 'ESP', 'EST', 'ETH', 'FIN', 'FJI', 'FRA',
       'FSM', 'GAB', 'GBR', 'GEO', 'GHA', 'GIN', 'GLP', 'GMB', 'GNB',
       'GNQ', 'GRC', 'GRD', 'GRL', 'GTM', 'GUF', 'GUM', 'GUY', 'HKG',
       'HND', 'HRV', 'HTI', 'HUN', 'IDN', 'IND', 'IRL', 'IRN', 'IRQ',
       'ISL', 'ISR', 'ITA', 'JAM', 'JOR', 'JPN', 'KAZ', 'KEN', 'KGZ',
       'KHM', 'KIR', 'KNA', 'KOR', 'KWT', 'LAO', 'LBN', 'LBR', 'LBY',
       'LCA', 'LKA', 'LSO', 'LTU', 'LUX', 'LVA', 'MAR', 'MCO', 'MDA',
       'MDG', 'MDV', 'MEX', 'MHL', 'MKD', 'MLI', 'MLT', 'MMR', 'MNE',
       'MNG', 'MNP', 'MOZ', 'MRT', 'MTQ', 'MUS', 'MWI', 'MYS', 'NAM',
       'NCL', 'NER', 'NGA', 'NIC', 'NIU', 'NLD', 'NOR', 'NPL', 'NRU',
       'NZL', 'OMN', 'OWID_CZS', 'OWID_KOS', 'OWID_USS', 'OWID_WRL',
       'OWID_YGS', 'PAK', 'PAN', 'PER', 'PHL', 'PLW', 'PNG', 'POL', 'PRI',
       'PRK', 'PRT', 'PRY', 'PSE', 'PYF', 'QAT', 'ROU', 'RUS', 'RWA',
       'SAU', 'SDN', 'SEN', 'SGP', 'SLB', 'SLE', 'SLV', 'SMR', 'SOM',
       'SRB', 'SSD', 'STP', 'SUR', 'SVK', 'SVN', 'SWE', 'SWZ', 'SYC',
       'SYR', 'TCD', 'TGO', 'THA', 'TJK', 'TKL', 'TKM', 'TLS', 'TON',
       'TTO', 'TUN', 'TUR', 'TUV', 'TWN', 'TZA', 'UGA', 'UKR', 'URY',
       'USA', 'UZB', 'VCT', 'VEN', 'VIR', 'VNM', 'VUT', 'WLF', 'WSM',
       'YEM', 'ZAF', 'ZMB', 'ZWE'], dtype=object)
```

In [41]:

```
cat_enc_le
```

Out[41]:

```
array([  0,   0,   0, ..., 216, 216, 216])
```

In [47]:

```
np.unique(cat_enc_le)
```

Out[47]:

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216])
```

In [43]:

```
# В этом примере видно, что перед кодированием
# уникальные значения признака сортируются в лексикографиеском порядке
le.inverse_transform([0, 1, 2, 3])
```

Out[43]:

```
array(['AFG', 'AGO', 'ALB', 'AND'], dtype=object)
```

In [44]:

```
from sklearn.preprocessing import OrdinalEncoder
```

In [49]:

```
data_oe = data[['Entity', 'Code']]
data_oe.head()
```

Out[49]:

| | Entity | Code |
|---|---|---|
| **0** | Afghanistan | AFG |
| **1** | Afghanistan | AFG |
| **2** | Afghanistan | AFG |
| **3** | Afghanistan | AFG |
| **4** | Afghanistan | AFG |

In [50]:

```
imp4 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='NA')
data_oe_filled = imp4.fit_transform(data_oe)
data_oe_filled
```

Out[50]:

```
array([['Afghanistan', 'AFG'],
       ['Afghanistan', 'AFG'],
       ['Afghanistan', 'AFG'],
       ...,
       ['Zimbabwe', 'ZWE'],
       ['Zimbabwe', 'ZWE'],
       ['Zimbabwe', 'ZWE']], dtype=object)
```

In [52]:

```
oe = OrdinalEncoder()
cat_enc_oe = oe.fit_transform(data_oe_filled)
cat_enc_oe
```

Out[52]:

```
array([[  0.,   0.],
       [  0.,   0.],
       [  0.,   0.],
       ...,
       [292., 217.],
       [292., 217.],
       [292., 217.]])
```

In [53]:

```
# Уникальные значения 1 признака
np.unique(cat_enc_oe[:, 0])
```

Out[53]:

```
array([  0.,   1.,   2.,   3.,   4.,   5.,   6.,   7.,   8.,   9.,  10.,
        11.,  12.,  13.,  14.,  15.,  16.,  17.,  18.,  19.,  20.,  21.,
        22.,  23.,  24.,  25.,  26.,  27.,  28.,  29.,  30.,  31.,  32.,
        33.,  34.,  35.,  36.,  37.,  38.,  39.,  40.,  41.,  42.,  43.,
        44.,  45.,  46.,  47.,  48.,  49.,  50.,  51.,  52.,  53.,  54.,
        55.,  56.,  57.,  58.,  59.,  60.,  61.,  62.,  63.,  64.,  65.,
        66.,  67.,  68.,  69.,  70.,  71.,  72.,  73.,  74.,  75.,  76.,
        77.,  78.,  79.,  80.,  81.,  82.,  83.,  84.,  85.,  86.,  87.,
        88.,  89.,  90.,  91.,  92.,  93.,  94.,  95.,  96.,  97.,  98.,
        99., 100., 101., 102., 103., 104., 105., 106., 107., 108., 109.,
       110., 111., 112., 113., 114., 115., 116., 117., 118., 119., 120.,
       121., 122., 123., 124., 125., 126., 127., 128., 129., 130., 131.,
       132., 133., 134., 135., 136., 137., 138., 139., 140., 141., 142.,
       143., 144., 145., 146., 147., 148., 149., 150., 151., 152., 153.,
       154., 155., 156., 157., 158., 159., 160., 161., 162., 163., 164.,
       165., 166., 167., 168., 169., 170., 171., 172., 173., 174., 175.,
       176., 177., 178., 179., 180., 181., 182., 183., 184., 185., 186.,
       187., 188., 189., 190., 191., 192., 193., 194., 195., 196., 197.,
       198., 199., 200., 201., 202., 203., 204., 205., 206., 207., 208.,
       209., 210., 211., 212., 213., 214., 215., 216., 217., 218., 219.,
       220., 221., 222., 223., 224., 225., 226., 227., 228., 229., 230.,
       231., 232., 233., 234., 235., 236., 237., 238., 239., 240., 241.,
       242., 243., 244., 245., 246., 247., 248., 249., 250., 251., 252.,
       253., 254., 255., 256., 257., 258., 259., 260., 261., 262., 263.,
       264., 265., 266., 267., 268., 269., 270., 271., 272., 273., 274.,
       275., 276., 277., 278., 279., 280., 281., 282., 283., 284., 285.,
       286., 287., 288., 289., 290., 291., 292.])
```

In [54]:

```
# Уникальные значения 2 признака
np.unique(cat_enc_oe[:, 1])
```

Out[54]:

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.,
       11., 12., 13., 14., 15., 16., 17., 18., 19., 20., 21.,
       22., 23., 24., 25., 26., 27., 28., 29., 30., 31., 32.,
       33., 34., 35., 36., 37., 38., 39., 40., 41., 42., 43.,
       44., 45., 46., 47., 48., 49., 50., 51., 52., 53., 54.,
       55., 56., 57., 58., 59., 60., 61., 62., 63., 64., 65.,
       66., 67., 68., 69., 70., 71., 72., 73., 74., 75., 76.,
       77., 78., 79., 80., 81., 82., 83., 84., 85., 86., 87.,
       88., 89., 90., 91., 92., 93., 94., 95., 96., 97., 98.,
       99., 100., 101., 102., 103., 104., 105., 106., 107., 108., 109.,
      110., 111., 112., 113., 114., 115., 116., 117., 118., 119., 120.,
      121., 122., 123., 124., 125., 126., 127., 128., 129., 130., 131.,
      132., 133., 134., 135., 136., 137., 138., 139., 140., 141., 142.,
      143., 144., 145., 146., 147., 148., 149., 150., 151., 152., 153.,
      154., 155., 156., 157., 158., 159., 160., 161., 162., 163., 164.,
      165., 166., 167., 168., 169., 170., 171., 172., 173., 174., 175.,
      176., 177., 178., 179., 180., 181., 182., 183., 184., 185., 186.,
      187., 188., 189., 190., 191., 192., 193., 194., 195., 196., 197.,
      198., 199., 200., 201., 202., 203., 204., 205., 206., 207., 208.,
      209., 210., 211., 212., 213., 214., 215., 216., 217.])
```

In [56]:

```
# Наименования категорий в соответствии с порядковыми номерами
oe.categories_
```

Out[56]:

```
[array(['Afghanistan', 'Africa', 'African Region', 'African Union',
       'Albania', 'Algeria', 'America', 'American Samoa',
       'Andean Latin America', 'Andorra', 'Angola', 'Antigua and Barbuda',
       'Argentina', 'Armenia', 'Asia', 'Australasia',
       'Australasia & Oceania', 'Australia', 'Austria', 'Azerbaijan',
       'Bahamas', 'Bahrain', 'Bangladesh', 'Barbados', 'Belarus',
       'Belgium', 'Belize', 'Benin', 'Bermuda', 'Bhutan', 'Bolivia',
       'Bosnia and Herzegovina', 'Bosnia-Herzegovina', 'Botswana',
       'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burundi',
       'Cambodia', 'Cameroon', 'Canada', 'Cape Verde', 'Caribbean',
       'Central African Republic', 'Central America & Caribbean',
       'Central Asia', 'Central Europe',
       'Central Europe, Eastern Europe, and Central Asia',
       'Central Latin America', 'Central sub-Saharan Africa', 'Chad',
       'Chile', 'China', 'Colombia', 'Commonwealth',
       'Commonwealth High Income', 'Commonwealth Low Income',
       'Commonwealth Middle Income', 'Comoros', 'Congo', 'Cook Islands',
       'Costa Rica', "Cote d'Ivoire", 'Croatia', 'Cuba', 'Cyprus',
       'Czechia', 'Czechoslovakia', 'Democratic Republic of Congo',
       'Denmark', 'Djibouti', 'Dominica', 'Dominican Republic',
       'East Asia', 'East Asia & Pacific - World Bank region',
       'East Germany (GDR)', 'East Timor', 'Eastern Europe',
       'Eastern Mediterranean Region', 'Eastern sub-Saharan Africa',
       'Ecuador', 'Egypt', 'El Salvador', 'England', 'Equatorial Guinea',
       'Eritrea', 'Estonia', 'Eswatini', 'Ethiopia', 'Europe',
       'Europe & Central Asia - World Bank region', 'European Region',
       'European Union', 'Fiji', 'Finland', 'France', 'French Guiana',
       'French Polynesia', 'G20', 'Gabon', 'Gambia', 'Georgia', 'Germany',
       'Ghana', 'Greece', 'Greenland', 'Grenada', 'Guadeloupe', 'Guam',
       'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti',
       'High SDI', 'High-income', 'High-income Asia Pacific',
       'High-income North America', 'High-middle SDI', 'Honduras',
       'Hong Kong', 'Hungary', 'Iceland', 'India', 'Indonesia',
       'International', 'Iran', 'Iraq', 'Ireland', 'Israel', 'Italy',
       'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
       'Kosovo', 'Kuwait', 'Kyrgyzstan', 'Laos',
       'Latin America & Caribbean - World Bank region', 'Latvia',
       'Lebanon', 'Lesotho', 'Liberia', 'Libya', 'Lithuania', 'Low SDI',
       'Low-middle SDI', 'Luxembourg', 'Macau', 'Madagascar', 'Malawi',
       'Malaysia', 'Maldives', 'Mali', 'Malta', 'Marshall Islands',
       'Martinique', 'Mauritania', 'Mauritius', 'Mexico',
       'Micronesia (country)', 'Middle East & North Africa', 'Middle SDI',
       'Moldova', 'Monaco', 'Mongolia', 'Montenegro', 'Morocco',
       'Mozambique', 'Myanmar', 'Namibia', 'Nauru', 'Nepal',
       'Netherlands', 'New Caledonia', 'New Zealand', 'Nicaragua',
       'Niger', 'Nigeria', 'Niue', 'Nordic Region',
       'North Africa and Middle East', 'North America', 'North Korea',
       'North Macedonia', 'Northern Ireland', 'Northern Mariana Islands',
       'Norway', 'OECD Countries', 'Oceania', 'Oman', 'Pakistan', 'Palau',
       'Palestine', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru',
       'Philippines', 'Poland', 'Portugal', 'Puerto Rico', 'Qatar',
       'Region of the Americas', 'Romania', 'Russia', 'Rwanda',
       'Saint Kitts and Nevis', 'Saint Lucia',
       'Saint Vincent and the Grenadines', 'Samoa', 'San Marino',
       'Sao Tome and Principe', 'Saudi Arabia', 'Scotland', 'Senegal',
       'Serbia', 'Serbia-Montenegro', 'Seychelles', 'Sierra Leone',
```

```
       'Singapore', 'Slovakia', 'Slovenia', 'Solomon Islands', 'Somalia',
       'South Africa', 'South America', 'South Asia',
       'South Asia - World Bank region', 'South Korea', 'South Sudan',
       'South-East Asia Region', 'Southeast Asia',
       'Southeast Asia, East Asia, and Oceania', 'Southern Latin America',
       'Southern sub-Saharan Africa', 'Spain', 'Sri Lanka',
       'Sub-Saharan Africa', 'Sub-Saharan Africa - World Bank region',
       'Sudan', 'Suriname', 'Sweden', 'Switzerland', 'Syria', 'Taiwan',
       'Tajikistan', 'Tanzania', 'Thailand', 'Timor', 'Togo', 'Tokelau',
       'Tonga', 'Trinidad and Tobago', 'Tropical Latin America',
       'Tunisia', 'Turkey', 'Turkmenistan', 'Tuvalu', 'USSR', 'Uganda',
       'Ukraine', 'United Arab Emirates', 'United Kingdom',
       'United States', 'United States Virgin Islands', 'Uruguay',
       'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam', 'Wales',
       'Wallis and Futuna', 'West Germany (FRG)', 'Western Europe',
       'Western Pacific Region', 'Western Sahara',
       'Western sub-Saharan Africa', 'World', 'World (excluding China)',
       'World Bank High Income', 'World Bank Low Income',
       'World Bank Lower Middle Income', 'World Bank Upper Middle Income',
       'Yemen', 'Yugoslavia', 'Zaire', 'Zambia', 'Zimbabwe'], dtype=object),
 array(['AFG', 'AGO', 'ALB', 'AND', 'ARE', 'ARG', 'ARM', 'ASM', 'ATG',
       'AUS', 'AUT', 'AZE', 'BDI', 'BEL', 'BEN', 'BFA', 'BGD', 'BGR',
       'BHR', 'BHS', 'BIH', 'BLR', 'BLZ', 'BMU', 'BOL', 'BRA', 'BRB',
       'BRN', 'BTN', 'BWA', 'CAF', 'CAN', 'CHE', 'CHL', 'CHN', 'CIV',
       'CMR', 'COD', 'COG', 'COK', 'COL', 'COM', 'CPV', 'CRI', 'CUB',
       'CYP', 'CZE', 'DEU', 'DJI', 'DMA', 'DNK', 'DOM', 'DZA', 'ECU',
       'EGY', 'ERI', 'ESH', 'ESP', 'EST', 'ETH', 'FIN', 'FJI', 'FRA',
       'FSM', 'GAB', 'GBR', 'GEO', 'GHA', 'GIN', 'GLP', 'GMB', 'GNB',
       'GNQ', 'GRC', 'GRD', 'GRL', 'GTM', 'GUF', 'GUM', 'GUY', 'HKG',
       'HND', 'HRV', 'HTI', 'HUN', 'IDN', 'IND', 'IRL', 'IRN', 'IRQ',
       'ISL', 'ISR', 'ITA', 'JAM', 'JOR', 'JPN', 'KAZ', 'KEN', 'KGZ',
       'KHM', 'KIR', 'KNA', 'KOR', 'KWT', 'LAO', 'LBN', 'LBR', 'LBY',
       'LCA', 'LKA', 'LSO', 'LTU', 'LUX', 'LVA', 'MAR', 'MCO', 'MDA',
       'MDG', 'MDV', 'MEX', 'MHL', 'MKD', 'MLI', 'MLT', 'MMR', 'MNE',
       'MNG', 'MNP', 'MOZ', 'MRT', 'MTQ', 'MUS', 'MWI', 'MYS', 'NA',
       'NAM', 'NCL', 'NER', 'NGA', 'NIC', 'NIU', 'NLD', 'NOR', 'NPL',
       'NRU', 'NZL', 'OMN', 'OWID_CZS', 'OWID_KOS', 'OWID_USS',
       'OWID_WRL', 'OWID_YGS', 'PAK', 'PAN', 'PER', 'PHL', 'PLW', 'PNG',
       'POL', 'PRI', 'PRK', 'PRT', 'PRY', 'PSE', 'PYF', 'QAT', 'ROU',
       'RUS', 'RWA', 'SAU', 'SDN', 'SEN', 'SGP', 'SLB', 'SLE', 'SLV',
       'SMR', 'SOM', 'SRB', 'SSD', 'STP', 'SUR', 'SVK', 'SVN', 'SWE',
       'SWZ', 'SYC', 'SYR', 'TCD', 'TGO', 'THA', 'TJK', 'TKL', 'TKM',
       'TLS', 'TON', 'TTO', 'TUN', 'TUR', 'TUV', 'TWN', 'TZA', 'UGA',
       'UKR', 'URY', 'USA', 'UZB', 'VCT', 'VEN', 'VIR', 'VNM', 'VUT',
       'WLF', 'WSM', 'YEM', 'ZAF', 'ZMB', 'ZWE'], dtype=object)]
```

In [57]:

```
# Обратное преобразование
oe.inverse_transform(cat_enc_oe)
```

Out[57]:

```
array([['Afghanistan', 'AFG'],
       ['Afghanistan', 'AFG'],
       ['Afghanistan', 'AFG'],
       ...,
       ['Zimbabwe', 'ZWE'],
       ['Zimbabwe', 'ZWE'],
       ['Zimbabwe', 'ZWE']], dtype=object)
```

# Проблемы использования LabelEncoder и OrdinalEncoder

Необходимо отметить, что LabelEncoder и OrdinalEncoder могут использоваться только для категориальных признаков в номинальных шкалах (для которых отсутствует порядок), например города, страны, названия рек и т.д.

Это связано с тем, что задать какой-либо порядок при кодировании с помощью LabelEncoder и OrdinalEncoder невозможно, они сортируют категории в лексикографическом порядке.

При этом кодирование целыми числами создает фиктивное отношение порядка (1 < 2 < 3 < ...) которого не было в исходных номинальных шкалах. Данное отношение порядка может негативно повлиять на построение модели машинного обучения.

# Масштабирование данных

Термины "масштабирование" и "нормализация" часто используются как синонимы, но это неверно. Масштабирование предполагает изменение диапазона измерения величины, а нормализация - изменение распределения этой величины. В этом разделе рассматривается только масштабирование.

Если признаки лежат в различных диапазонах, то необходимо их нормализовать. Как правило, применяют два подхода:

MinMax масштабирование: хновый=хстарый−min(X)max(X)−min(X) В этом случае значения лежат в диапазоне от 0 до 1.

Масштабирование данных на основе Z-оценки: хновый=хстарый−AVG(X)σ(X) В этом случае большинство значений попадает в диапазон от -3 до 3.

где X - матрица объект-признак, AVG(X) - среднее значение, σ - среднеквадратичное отклонение.

In [58]:

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

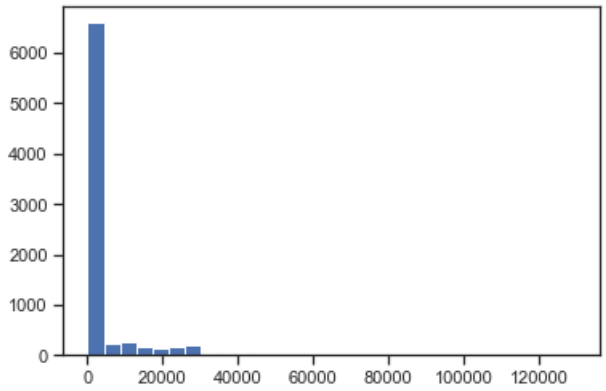
# MinMax масштабирование

In [67]:

```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['Deaths - Fire, heat, and hot substances - Sex: Both - Age: All Ages (Number)']])
```

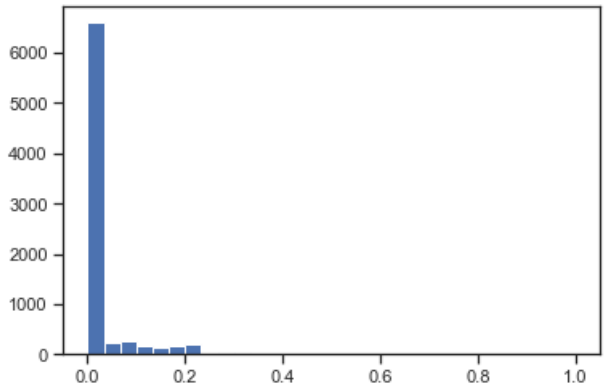
In [68]:

```
plt.hist(data['Deaths - Fire, heat, and hot substances - Sex: Both - Age: All Ages (Number)'], 30)
plt.show()
```




In [70]:

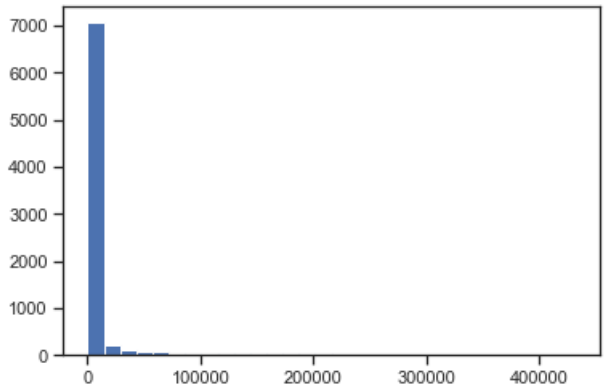
```
plt.hist(sc1_data, 30)
plt.show()
```




# Масштабирование данных на основе Z-оценки - StandardScaler

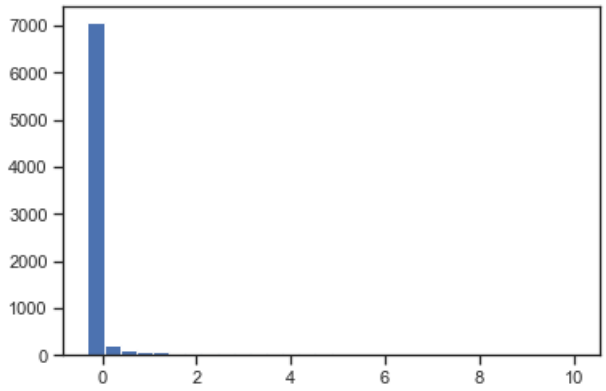
In [77]:

```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Deaths - Meningitis - Sex: Both - Age: All Ages (Number)']])
plt.hist(data['Deaths - Meningitis - Sex: Both - Age: All Ages (Number)'], 30)
plt.show()
```




In [73]:

```
plt.hist(sc2_data, 30)
plt.show()
```




In [ ]: