In [1]:

```
!pip install jupyterthemes
!jt -t chesterish
```

```
Requirement already satisfied: jupyterthemes in c:\programdata\anaconda3\lib\site-packag
es (0.20.0)
Requirement already satisfied: jupyter-core in c:\users\nikita\appdata\roaming\python\py
thon38\site-packages (from jupyterthemes) (4.7.1)
Requirement already satisfied: matplotlib>=1.4.3 in c:\programdata\anaconda3\lib\site-pa
ckages (from jupyterthemes) (3.3.2)
Requirement already satisfied: notebook>=5.6.0 in c:\programdata\anaconda3\lib\site-pack
ages (from jupyterthemes) (6.1.4)
Requirement already satisfied: lesscpy>=0.11.2 in c:\programdata\anaconda3\lib\site-pack
ages (from jupyterthemes) (0.14.0)
Requirement already satisfied: ipython>=5.4.1 in c:\users\nikita\appdata\roaming\python
\python38\site-packages (from jupyterthemes) (7.22.0)
Requirement already satisfied: pywin32>=1.0; sys_platform == "win32" in c:\users\nikita
\appdata\roaming\python\python38\site-packages (from jupyter-core->jupyterthemes) (300)
Requirement already satisfied: traitlets in c:\users\nikita\appdata\roaming\python\pytho
n38\site-packages (from jupyter-core->jupyterthemes) (5.0.5)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-pa
ckages (from matplotlib>=1.4.3->jupyterthemes) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-package
s (from matplotlib>=1.4.3->jupyterthemes) (0.10.0)
Requirement already satisfied: certifi>=2020.06.20 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib>=1.4.3->jupyterthemes) (2020.6.20)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\nikita\appdata\roaming\p
ython\python38\site-packages (from matplotlib>=1.4.3->jupyterthemes) (2.8.1)
Requirement already satisfied: numpy>=1.15 in c:\programdata\anaconda3\lib\site-packages
(from matplotlib>=1.4.3->jupyterthemes) (1.19.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\programdat
a\anaconda3\lib\site-packages (from matplotlib>=1.4.3->jupyterthemes) (2.4.7)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packag
es (from matplotlib>=1.4.3->jupyterthemes) (8.0.1)
Requirement already satisfied: ipython-genutils in c:\users\nikita\appdata\roaming\pytho
n\python38\site-packages (from notebook>=5.6.0->jupyterthemes) (0.2.0)
Requirement already satisfied: jupyter-client>=5.3.4 in c:\users\nikita\appdata\roaming
\python\python38\site-packages (from notebook>=5.6.0->jupyterthemes) (6.1.12)
Requirement already satisfied: jinja2 in c:\programdata\anaconda3\lib\site-packages (fro
m notebook>=5.6.0->jupyterthemes) (2.11.2)
Requirement already satisfied: nbconvert in c:\programdata\anaconda3\lib\site-packages
(from notebook>=5.6.0->jupyterthemes) (6.0.7)
Requirement already satisfied: nbformat in c:\programdata\anaconda3\lib\site-packages (f
rom notebook>=5.6.0->jupyterthemes) (5.0.8)
Requirement already satisfied: ipykernel in c:\users\nikita\appdata\roaming\python\pytho
n38\site-packages (from notebook>=5.6.0->jupyterthemes) (5.5.0)
Requirement already satisfied: prometheus-client in c:\programdata\anaconda3\lib\site-pa
ckages (from notebook>=5.6.0->jupyterthemes) (0.8.0)
Requirement already satisfied: terminado>=0.8.3 in c:\programdata\anaconda3\lib\site-pac
kages (from notebook>=5.6.0->jupyterthemes) (0.9.1)
Requirement already satisfied: Send2Trash in c:\programdata\anaconda3\lib\site-packages
(from notebook>=5.6.0->jupyterthemes) (1.5.0)
Requirement already satisfied: tornado>=5.0 in c:\users\nikita\appdata\roaming\python\py
thon38\site-packages (from notebook>=5.6.0->jupyterthemes) (6.1)
Requirement already satisfied: argon2-cffi in c:\programdata\anaconda3\lib\site-packages
(from notebook>=5.6.0->jupyterthemes) (20.1.0)
Requirement already satisfied: pyzmq>=17 in c:\users\nikita\appdata\roaming\python\pytho
n38\site-packages (from notebook>=5.6.0->jupyterthemes) (22.0.3)
Requirement already satisfied: six in c:\users\nikita\appdata\roaming\python\python38\si
te-packages (from lesscpy>=0.11.2->jupyterthemes) (1.14.0)
Requirement already satisfied: ply in c:\programdata\anaconda3\lib\site-packages (from l
esscpy>=0.11.2->jupyterthemes) (3.11)
Requirement already satisfied: pickleshare in c:\users\nikita\appdata\roaming\python\pyt
hon38\site-packages (from ipython>=5.4.1->jupyterthemes) (0.7.5)
Requirement already satisfied: decorator in c:\users\nikita\appdata\roaming\python\pytho
n38\site-packages (from ipython>=5.4.1->jupyterthemes) (4.4.2)
Requirement already satisfied: setuptools>=18.5 in c:\programdata\anaconda3\lib\site-pac
kages (from ipython>=5.4.1->jupyterthemes) (50.3.1.post20201107)
```

```
Requirement already satisfied: backcall in c:\users\nikita\appdata\roaming\python\python38\site-packages (from ipython>=5.4.1->jupyterthemes) (0.2.0)
Requirement already satisfied: jedi>=0.16 in c:\users\nikita\appdata\roaming\python\python38\site-packages (from ipython>=5.4.1->jupyterthemes) (0.18.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in c:\users\nikita\appdata\roaming\python\python38\site-packages (from ipython>=5.4.1->jupyterthemes) (3.0.18)
Requirement already satisfied: pygments in c:\users\nikita\appdata\roaming\python\python38\site-packages (from ipython>=5.4.1->jupyterthemes) (2.8.1)
Requirement already satisfied: colorama; sys_platform == "win32" in c:\users\nikita\appdata\roaming\python\python38\site-packages (from ipython>=5.4.1->jupyterthemes) (0.4.3)
Requirement already satisfied: MarkupSafe>=0.23 in c:\programdata\anaconda3\lib\site-packages (from jinja2->notebook>=5.6.0->jupyterthemes) (1.1.1)
Requirement already satisfied: entrypoints>=0.2.2 in c:\programdata\anaconda3\lib\site-packages (from nbconvert->notebook>=5.6.0->jupyterthemes) (0.3)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert->notebook>=5.6.0->jupyterthemes) (0.5.1)
Requirement already satisfied: jupyterlab-pygments in c:\programdata\anaconda3\lib\site-packages (from nbconvert->notebook>=5.6.0->jupyterthemes) (0.1.2)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert->notebook>=5.6.0->jupyterthemes) (1.4.3)
Requirement already satisfied: testpath in c:\programdata\anaconda3\lib\site-packages (from nbconvert->notebook>=5.6.0->jupyterthemes) (0.4.4)
Requirement already satisfied: bleach in c:\programdata\anaconda3\lib\site-packages (from nbconvert->notebook>=5.6.0->jupyterthemes) (3.2.1)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert->notebook>=5.6.0->jupyterthemes) (0.8.4)
Requirement already satisfied: defusedxml in c:\programdata\anaconda3\lib\site-packages (from nbconvert->notebook>=5.6.0->jupyterthemes) (0.6.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\programdata\anaconda3\lib\site-packages (from nbformat->notebook>=5.6.0->jupyterthemes) (3.2.0)
Requirement already satisfied: pywinpty>=0.5 in c:\programdata\anaconda3\lib\site-packages (from terminado>=0.8.3->notebook>=5.6.0->jupyterthemes) (0.5.7)
Requirement already satisfied: cffi>=1.0.0 in c:\programdata\anaconda3\lib\site-packages (from argon2-cffi->notebook>=5.6.0->jupyterthemes) (1.14.3)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\nikita\appdata\roaming\python\python38\site-packages (from jedi>=0.16->ipython>=5.4.1->jupyterthemes) (0.8.1)
Requirement already satisfied: wcwidth in c:\programdata\anaconda3\lib\site-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=5.4.1->jupyterthemes) (0.2.5)
Requirement already satisfied: nest-asyncio in c:\programdata\anaconda3\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert->notebook>=5.6.0->jupyterthemes) (1.4.2)
Requirement already satisfied: async-generator in c:\programdata\anaconda3\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert->notebook>=5.6.0->jupyterthemes) (1.10)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from bleach->nbconvert->notebook>=5.6.0->jupyterthemes) (20.4)
Requirement already satisfied: webencodings in c:\programdata\anaconda3\lib\site-packages (from bleach->nbconvert->notebook>=5.6.0->jupyterthemes) (0.5.1)
Requirement already satisfied: attrs>=17.4.0 in c:\programdata\anaconda3\lib\site-packages (from jsonschema!=2.5.0,>=2.4->nbformat->notebook>=5.6.0->jupyterthemes) (20.3.0)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\programdata\anaconda3\lib\site-packages (from jsonschema!=2.5.0,>=2.4->nbformat->notebook>=5.6.0->jupyterthemes) (0.17.3)
Requirement already satisfied: pycparser in c:\programdata\anaconda3\lib\site-packages (from cffi>=1.0.0->argon2-cffi->notebook>=5.6.0->jupyterthemes) (2.20)
```

Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных.

Зачем обрабатывать пропуски?

- Если в данных есть пропуски, то большинство алгоритмов машинного обучения не будут с ними работать. Даже корреляционная матрица не будет строиться корректно.

- Большинство алгоритмов машинного обучения требуют явного перекодирования категориальных признаков в числовые. Даже если алгоритм не требует этого явно, такое перекодирование возможно стоит попробовать, чтобы повысить качество модели.
- Большинство алгоритмов показывает лучшее качество на масштабированных признаках, в особенности алгоритмы, использующие методы градиентного спуска.

1) Загрузка, импорт и первичный анализ данных

Импорт библиотек

Импортируем библиотеки с помощью команды import. Будем подключать все библиотеки последовательно, по мере их использования.

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Используем данные из прошлой лабораторной работы, а именно vgsales.csv

In [3]:
```
data = pd.read_csv('database/vgsales.csv', sep = ",")
```

In [4]:
```
# размер набора данных
print(f'Строк - {data.shape[0]}\nСтолбцов - {data.shape[1]}')
```

```
Строк - 16598
Столбцов - 11
```

In [5]:
```
# типы колонок
data.dtypes
```

Out[5]:
```
Rank              int64
Name             object
Platform         object
Year            float64
Genre            object
Publisher        object
NA_Sales        float64
EU_Sales        float64
JP_Sales        float64
Other_Sales     float64
Global_Sales    float64
dtype: object
```

In [6]:
```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

Out[6]:
```
Rank              0
Name              0
```

```
Platform           0
Year             271
Genre              1
Publisher         59
NA_Sales           0
EU_Sales           0
JP_Sales           0
Other_Sales        0
Global_Sales       0
dtype: int64
```

In [7]:
```
# Первые 5 строк датасета
data.head(6)
```

Out[7]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_S
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	8
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	4
2	3	Mario Kart Wii	Wii	2008.0	NaN	NaN	15.85	12.88	3.79	3.31	3
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	3
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	3
5	6	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22	0.58	3

Обработка пропусков в данных

Простые стратегии - удалениее или заполнение нулями

Удаление колонок, содержащих пустые значения `res = data.dropna(axis=1, how='any')`

Удаление строк, содержащих пустые значения `res = data.dropna(axis=0, how='any')`

На всякий случай вот документация

Удаление может производиться для группы строк или колонок.

In [8]:
```
# Удаление колонок, содержащих пустые значения
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

Out[8]: `((16598, 11), (16598, 8))`

In [9]:
```
# Удаление строк, содержащих пустые значения
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

Out[9]: `((16598, 11), (16290, 11))`

In [10]:
```python
data.head()
```

Out[10]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_S
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	8
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	4
2	3	Mario Kart Wii	Wii	2008.0	NaN	NaN	15.85	12.88	3.79	3.31	3
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	3
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	3

In [11]:
```python
# В данном случае это некорректно, так как нулями заполняются в том числе категориальные колонки
data_new_3 = data.fillna(0)
data_new_3.head(6)
```

Out[11]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_S
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	8
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	4
2	3	Mario Kart Wii	Wii	2008.0	0	0	15.85	12.88	3.79	3.31	3
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	3
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	3
5	6	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22	0.58	3

"Внедрение значений" - импьютация (imputation)

Обработка пропусков в числовых данных

In [12]:
```python
# 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]
    #print(f'temp_null_count {temp_null_count}')
    dt = str(data[col].dtype)
    #print(f'dt {dt}')
```

```
    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(f'Колонка {col}. Тип данных {dt}. Количество пустых значений {temp_null_count}, {temp_perc}%.')
```

```
Колонка Year. Тип данных float64. Количество пустых значений 271, 1.63%.
```

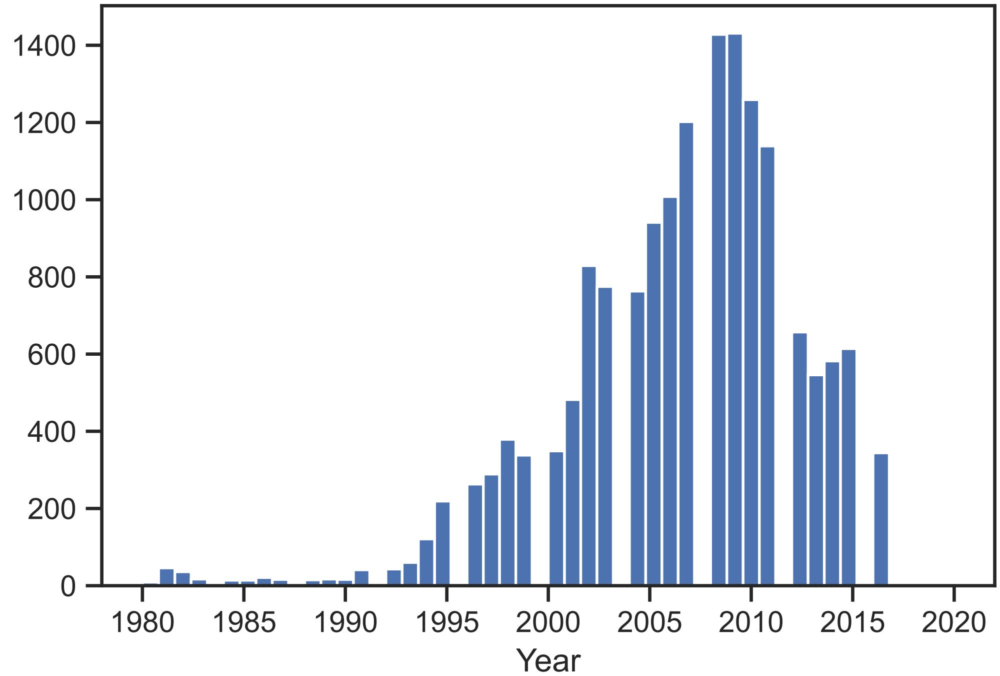
In [13]:
```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num
```

Out[13]:

	Year
0	2006.0
1	1985.0
2	2008.0
3	2009.0
4	1996.0
...	...
16593	2002.0
16594	2003.0
16595	2008.0
16596	2010.0
16597	2003.0

16598 rows × 1 columns

In [14]:
```
# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```



Будем использовать встроенные средства импьютации библиотеки scikit-learn - ссылка

In [15]:
```
data_num_MasVnrArea = data_num[['Year']]
data_num_MasVnrArea.head()
```

Out[15]:

	Year
0	2006.0
1	1985.0
2	2008.0
3	2009.0
4	1996.0

In [16]:
```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

In [17]:
```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_MasVnrArea)
mask_missing_values_only
```

Out[17]:
```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

С помощью класса SimpleImputer можно проводить импьютацию различными показателями центра распределения

In [18]:

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

In [19]:

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

In [20]:

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

Out[20]:

```
('mean',
 array([2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
        2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
```

```
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331, 2006.40644331,
         2006.40644331, 2006.40644331, 2006.40644331]))
```

In [21]:

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

Out[21]:

```
('median',
 array([2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007., 2007.,
        2007.]))
```

In [22]:

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

Out[22]:

```
('most_frequent',
 array([2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
        2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
```

```
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009., 2009.,
       2009.]))
```

In [23]:

```python
# 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 [24]:

```python
data[['Year']].describe()
```

Out[24]:

	Year
count	16327.000000
mean	2006.406443
std	5.828981
min	1980.000000
25%	2003.000000
50%	2007.000000
75%	2010.000000
max	2020.000000

In [25]:

```python
test_num_impute_col(data, 'Year', strategies[0])
```

Out[25]:

```
('Year', 'mean', 271, 2006.4064433147546, 2006.4064433147546)
```

In [26]:
```
test_num_impute_col(data, 'Year', strategies[1])
```
Out[26]:
```
('Year', 'median', 271, 2007.0, 2007.0)
```
In [27]:
```
test_num_impute_col(data, 'Year', strategies[2])
```
Out[27]:
```
('Year', 'most_frequent', 271, 2009.0, 2009.0)
```

Обработка пропусков в категориальных данных

In [28]:
```
# 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 / data.shape[0]) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {},
{}%.'.format(col, dt, temp_null_count, temp_perc))
```
```
Колонка Genre. Тип данных object. Количество пустых значений 1, 0.01%.
Колонка Publisher. Тип данных object. Количество пустых значений 59, 0.36%.
```

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

In [29]:
```
cat_temp_data = data[['Publisher']]
cat_temp_data.head()
```
Out[29]:

	Publisher
0	Nintendo
1	Nintendo
2	NaN
3	Nintendo
4	Nintendo

In [30]:
```
# Будем выводить только первые 10, т.к там уникальных значений свыше 1000
cat_temp_data['Publisher'].unique()[0:10]
```
Out[30]:
```
array(['Nintendo', nan, 'Microsoft Game Studios', 'Take-Two Interactive',
       'Sony Computer Entertainment', 'Activision', 'Ubisoft',
       'Bethesda Softworks', 'Electronic Arts', 'Sega'], dtype=object)
```
In [31]:
```
cat_temp_data[cat_temp_data['Publisher'].isnull()].shape
```

Out[31]: (59, 1)

In [32]:
```
# Импьютация наиболее частыми значениями
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

Out[32]:
```
array([['Nintendo'],
       ['Nintendo'],
       ['Electronic Arts'],
       ...,
       ['Activision'],
       ['7G//AMES'],
       ['Wanadoo']], dtype=object)
```

In [33]:
```
# Будем выводить только первые 10, т.к там уникальных значений свыше 1000
# Пустые значения отсутствуют
np.unique(data_imp2)[0:10]
```

Out[33]:
```
array(['10TACLE Studios', '1C Company', '20th Century Fox Video Games',
       '2D Boy', '3DO', '49Games', '505 Games', '5pb', '7G//AMES',
       '989 Sports'], dtype=object)
```

In [34]:
```
# Импьютация константой
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant',
fill_value='NA')
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

Out[34]:
```
array([['Nintendo'],
       ['Nintendo'],
       ['NA'],
       ...,
       ['Activision'],
       ['7G//AMES'],
       ['Wanadoo']], dtype=object)
```

In [35]:
```
# Будем выводить только первые 10, т.к там уникальных значений свыше 1000
np.unique(data_imp3)[0:10]
```

Out[35]:
```
array(['10TACLE Studios', '1C Company', '20th Century Fox Video Games',
       '2D Boy', '3DO', '49Games', '505 Games', '5pb', '7G//AMES',
       '989 Sports'], dtype=object)
```

In [36]:
```
data_imp3[data_imp3=='NA'].size
```

Out[36]: 59

Преобразование категориальных признаков в числовые

In [37]:
```
cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
cat_enc
```

Out[37]:

	c1
0	Nintendo
1	Nintendo
2	Electronic Arts
3	Nintendo
4	Nintendo
...	...
16593	Kemco
16594	Infogrames
16595	Activision
16596	7G//AMES
16597	Wanadoo

16598 rows × 1 columns

Кодирование категорий целочисленными значениями - label encoding

In [38]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [39]:
```
le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
```

In [40]:
```
cat_enc['c1'].unique()[0:10]
```

Out[40]:
```
array(['Nintendo', 'Electronic Arts', 'Microsoft Game Studios',
       'Take-Two Interactive', 'Sony Computer Entertainment',
       'Activision', 'Ubisoft', 'Bethesda Softworks', 'Sega',
       'SquareSoft'], dtype=object)
```

In [41]:
```
np.unique(cat_enc_le)
```

Out[41]:
```
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, 293, 294, 295, 296, 297, 298,
       299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311,
       312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324,
       325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337,
       338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350,
       351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363,
       364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376,
       377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389,
       390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402,
       403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415,
       416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428,
       429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441,
       442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454,
       455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467,
       468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480,
       481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493,
       494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506,
       507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519,
       520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532,
       533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545,
       546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558,
       559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571,
       572, 573, 574, 575, 576, 577])
```

In [42]:

```
le.inverse_transform([0, 1, 2, 3])
```

Out[42]:

```
array(['10TACLE Studios', '1C Company', '20th Century Fox Video Games',
       '2D Boy'], dtype=object)
```

Кодирование категорий наборами бинарных значений - one-hot encoding

In [43]:

```
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])
```

In [44]:

```
cat_enc.shape
```

Out[44]:

```
(16598, 1)
```

In [45]:

```
cat_enc_ohe.shape
```

Out[45]:

```
(16598, 578)
```

In [46]:

```
cat_enc_ohe
```

Out[46]:

```
<16598x578 sparse matrix of type '<class 'numpy.float64'>'
	with 16598 stored elements in Compressed Sparse Row format>
```

In [47]:

```
cat_enc_ohe.todense()[0:10]
```

Out[47]:

```
matrix([[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.]])
```

In [48]:
```
cat_enc.head(10)
```

Out[48]:

	c1
0	Nintendo
1	Nintendo
2	Electronic Arts
3	Nintendo
4	Nintendo
5	Nintendo
6	Nintendo
7	Nintendo
8	Nintendo
9	Nintendo

Pandas get_dummies - быстрый вариант one-hot кодирования

In [49]:
```
pd.get_dummies(cat_enc).head()
```

Out[49]:

	c1_10TACLE Studios	c1_1C Company	c1_20th Century Fox Video Games	c1_2D Boy	c1_3DO	c1_49Games	c1_505 Games	c1_5pb	c1_7G//AMES	c1_989 Sports	...	c1_ G
0	0	0	0	0	0	0	0	0	0	0	...	
1	0	0	0	0	0	0	0	0	0	0	...	
2	0	0	0	0	0	0	0	0	0	0	...	
3	0	0	0	0	0	0	0	0	0	0	...	
4	0	0	0	0	0	0	0	0	0	0	...	

5 rows × 578 columns

In [50]:
```
pd.get_dummies(cat_temp_data, dummy_na=True).head()
```

Out[50]:

	Publisher_10TACLE Studios	Publisher_1C Company	Publisher_20th Century Fox Video Games	Publisher_2D Boy	Publisher_3DO	Publisher_49Games	Publisher_5(Gam
0	0	0	0	0	0	0	
1	0	0	0	0	0	0	
2	0	0	0	0	0	0	
3	0	0	0	0	0	0	
4	0	0	0	0	0	0	

5 rows × 579 columns

Масштабирование данных

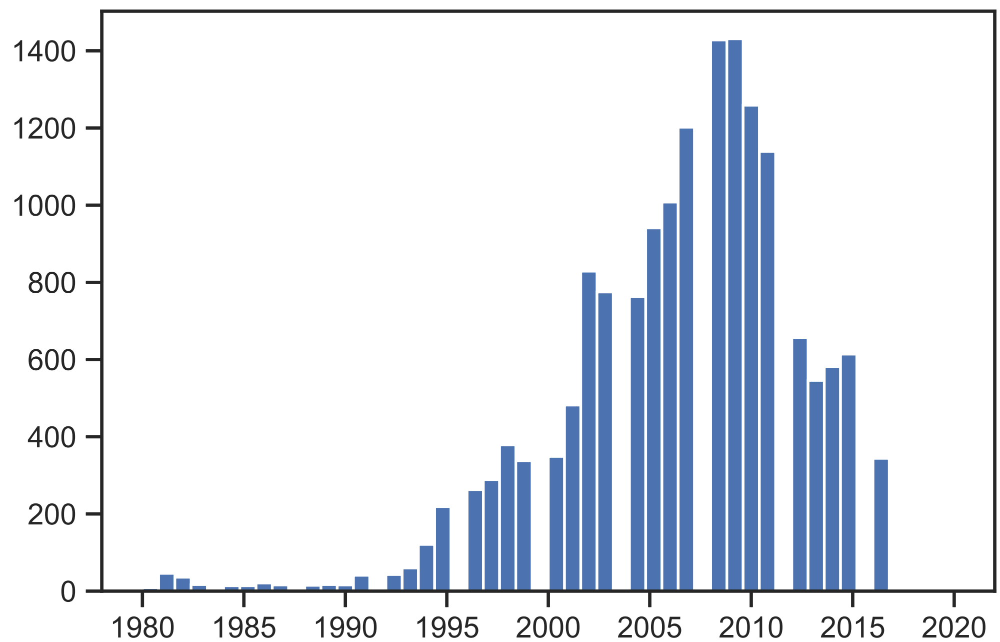
Термины "масштабирование" и "нормализация" часто используются как синонимы. Масштабирование предполагает изменение диапазона измерения величины, а нормализация - изменение распределения этой величины.

In [51]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

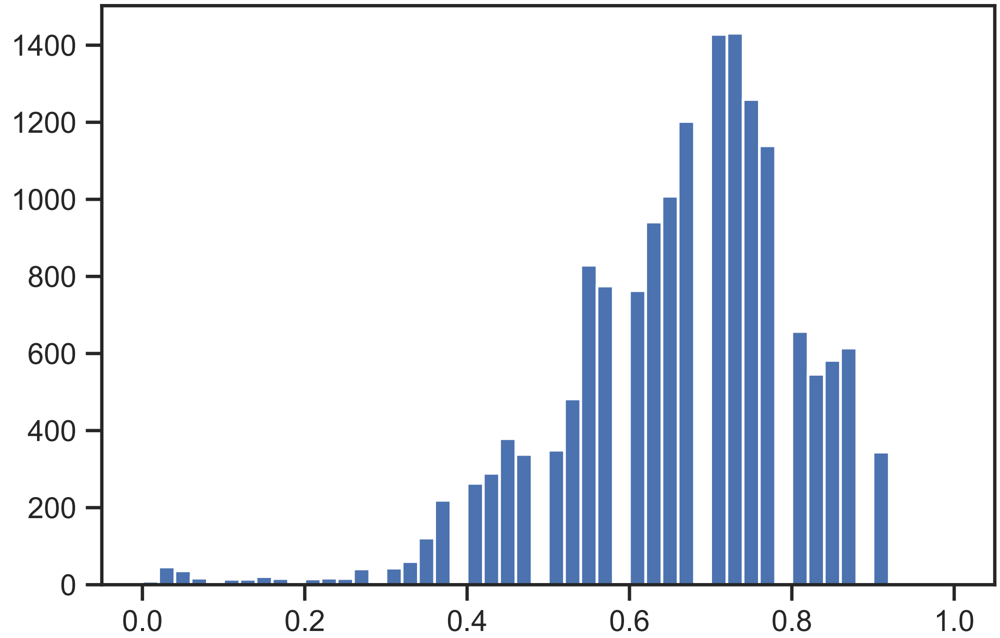
MinMax масштабирование

In [52]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['Year']])
```

In [53]:
```
plt.hist(data['Year'], 50)
plt.show()
```



In [54]:
```
plt.hist(sc1_data, 50)
plt.show()
```



Масштабирование данных на основе Z-оценки - StandardScaler

In [55]:
```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Year']])
```

In [56]:
```
plt.hist(sc2_data, 50)
plt.show()
```