Цель лабораторной работы: изучение способов предварительной обработки данных для дальнейшего формирования моделей. Задание: Выбрать набор данных (датасет), содержащий категориальные признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.) Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи: обработку пропусков в данных; кодирование категориальных признаков; масштабирование данных.

В чем состоит проблема: 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 отмасштабированных признаках, в особенности алгоритмы, использующие методы градиентного спуска.

In [3]:

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

companies = pd.read_csv('Data/lab_3/acquisitions.csv', sep=',')
companies.head(10)
```

Out[3]:

	AcquisitionID	AcquisitionMonth	AcquisitionMonthDate	AcquisitionYear	Company	
0	ACQ99	November	11.0	2015	bebop	Clou
1	ACQ98	November	11.0	2015	Fly Labs	Vi
2	ACQ97	December	8.0	2015	Clearleap	Cl ma
3	ACQ96	December	18.0	2015	Metanautix	
4	ACQ95	December	21.0	2015	Talko, Inc.	comn
5	ACQ94	January	7.0	2016	Emotient	
6	ACQ93	January	15.0	2016	Iris Analytics	frau
7	ACQ92	January	19.0	2016	Teacher Gaming LLC	
8	ACQ915	July	30.0	1987	Forethought, Inc.	
9	ACQ914	March	2.0	1988	Network Innovations	

In [4]:

```
companies.shape
```

Out[4]:

```
(916, 10)
```

In [5]:

```
companies.dtypes
```

Out[5]:

```
AcquisitionID             object
AcquisitionMonth          object
AcquisitionMonthDate     float64
AcquisitionYear            int64
Company                   object
Business                  object
Country                   object
Value (USD)              float64
Derived products          object
ParentCompany             object
dtype: object
```

In [6]:

```
# Проверка на пустые значения
companies.isnull().sum()
# for column in companies.columns:
#     buf_null = companies[companies[column].isnull()].shape[0]
#     print ('{}-{}'.format(column, buf_null))

# acquisition - приобретение, овладение
# derived products - производные продукты
```

Out[6]:

```
AcquisitionID             0
AcquisitionMonth          6
AcquisitionMonthDate     33
AcquisitionYear           0
Company                   0
Business                  0
Country                  46
Value (USD)             671
Derived products        515
ParentCompany             0
dtype: int64
```

In [7]:

```
#Вывод: по полям AcquisitionMont, AcquisitionMonthDate, Country-46 - пропуски данных не большие,
# Это не сильно повлияет на анализ
# По полям Value (USD) и Derived products пропуски более 50% от dataset,сильное влияние
total_count = companies.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 916
```

In [8]:

```
#1. Обработка пропусков в данных
#1.1. Простые стратегии - удаление или заполнение нулями
# Удаление колонок, содержащих пустые значения
data_new_1 = companies.dropna(axis=1, how='any')
(companies.shape, data_new_1.shape)
```

Out[8]:

```
((916, 10), (916, 5))
```

In [9]:

```
data_new_1.head(5)
```

Out[9]:

	AcquisitionID	AcquisitionYear	Company	Business	ParentCompany
0	ACQ99	2015	bebop	Cloud software	Google
1	ACQ98	2015	Fly Labs	Video editing	Google
2	ACQ97	2015	Clearleap	Cloud-based video management	IBM
3	ACQ96	2015	Metanautix	Big Data Analytics	Microsoft
4	ACQ95	2015	Talko, Inc.	Mobile communications	Microsoft

In [10]:

```
data_new_1.shape
```

Out[10]:

```
(916, 5)
```

In [11]:

```
# Удаление строк, содержащих пустые значения
data_new_2 = companies.dropna(axis=0, how='any')
(companies.shape, data_new_2.shape)
```

Out[11]:

```
((916, 10), (114, 10))
```

In [12]:

```
data_new_2.head(5)
```

Out[12]:

	AcquisitionID	AcquisitionMonth	AcquisitionMonthDate	AcquisitionYear	Company	Busi
0	ACQ99	November	11.0	2015	bebop	(sof
38	ACQ889	February	7.0	1997	NeXT	Uni harc sof plε
47	ACQ880	October	8.0	1997	Four11	Web-k
55	ACQ873	June	8.0	1998	Viaweb	applic
56	ACQ872	July	17.0	1998	Webcal	Calenc sof

In [13]:

```
data_new_2.shape
```

Out[13]:

```
(114, 10)
```

In [14]:

```
# Заполнение всех пропущенных значений нулями
# В данном случае это некорректно, так как нулями заполняются в том числе категориальны
е колонки
data_new_3 = companies.fillna(0)
data_new_3.isnull().sum()
```

Out[14]:

```
AcquisitionID           0
AcquisitionMonth        0
AcquisitionMonthDate    0
AcquisitionYear         0
Company                 0
Business                0
Country                 0
Value (USD)             0
Derived products        0
ParentCompany           0
dtype: int64
```

In [15]:

```
#1.2. "Внедрение значений" - импьютация (imputation)
#1.2.1. Обработка пропусков в числовых данных
# Импьютация - процесс замены пропущенных, некорректных или несостоятельных значений др
угими значениями
# Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
# Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
for col in companies.columns:
    # Количество пустых значений
    temp_null_count = companies[companies[col].isnull()].shape[0]
    dt = str(companies[col].dtype)
    total_count = companies.shape[0]
    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(c
ol, dt, temp_null_count, temp_perc))
```

```
Колонка AcquisitionMonthDate. Тип данных float64. Количество пустых значен
ий 33, 3.6%.
Колонка Value (USD). Тип данных float64. Количество пустых значений 671, 7
3.25%.
```

In [16]:

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

Out[16]:

	AcquisitionMonthDate	Value (USD)
0	11.0	3.800000e+08
1	11.0	NaN
2	8.0	NaN
3	18.0	NaN
4	21.0	NaN
5	7.0	NaN
6	15.0	NaN
7	19.0	NaN
8	30.0	1.400000e+07
9	2.0	NaN
10	7.0	NaN
11	27.0	NaN
12	11.0	NaN
13	3.0	NaN
14	21.0	NaN
15	31.0	NaN
16	29.0	NaN
17	28.0	NaN
18	27.0	NaN
19	1.0	NaN
20	15.0	NaN
21	23.0	NaN
22	10.0	NaN
23	17.0	NaN
24	6.0	NaN
25	28.0	NaN
26	16.0	NaN
27	12.0	NaN
28	16.0	1.330000e+08
29	6.0	NaN
...	...	...
886	23.0	NaN
887	31.0	1.600000e+08
888	3.0	NaN
889	6.0	1.000000e+09
890	NaN	NaN
891	5.0	NaN

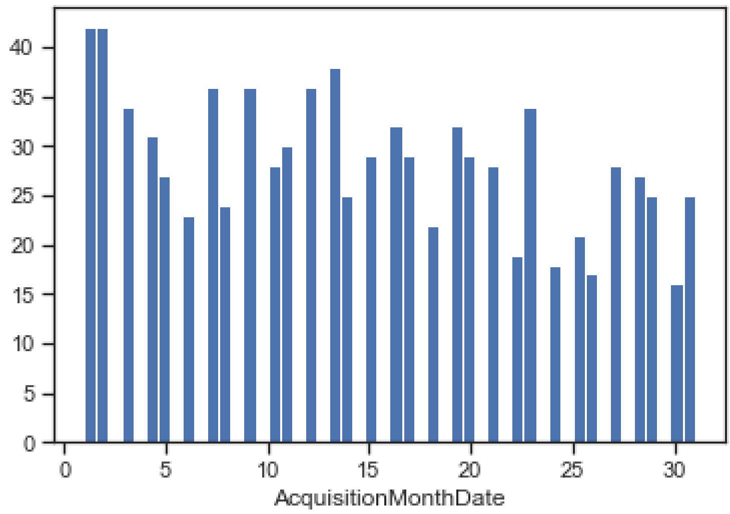
	AcquisitionMonthDate	Value (USD)
892	NaN	NaN
893	NaN	NaN
894	3.0	NaN
895	10.0	NaN
896	11.0	NaN
897	21.0	NaN
898	28.0	NaN
899	28.0	NaN
900	30.0	NaN
901	2.0	NaN
902	9.0	NaN
903	3.0	NaN
904	17.0	NaN
905	21.0	NaN
906	21.0	NaN
907	28.0	NaN
908	NaN	NaN
909	3.0	NaN
910	5.0	NaN
911	6.0	1.309000e+09
912	9.0	NaN
913	11.0	NaN
914	18.0	NaN
915	4.0	7.500000e+09

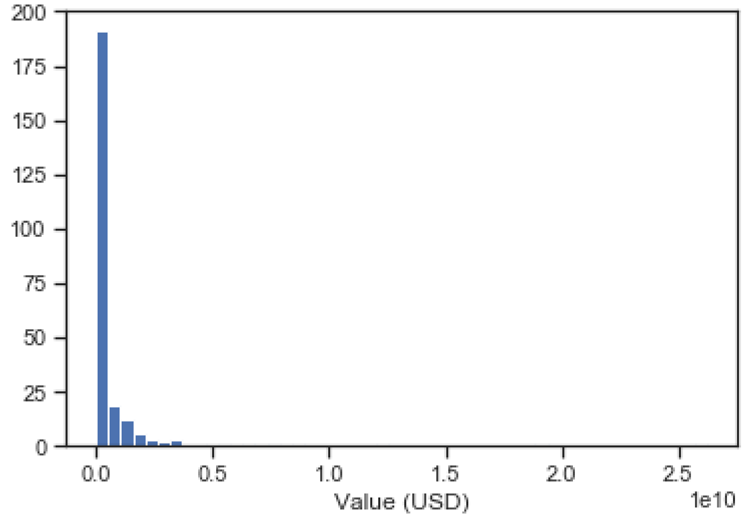
916 rows × 2 columns

In [17]:

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

```
c:\users\uivan_000\anaconda3\lib\site-packages\numpy\lib\histograms.py:82
4: RuntimeWarning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
c:\users\uivan_000\anaconda3\lib\site-packages\numpy\lib\histograms.py:82
5: RuntimeWarning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```





In [18]:

```
# Фильтр по пустым значениям поля AcquisitionMonthDate
companies[companies['AcquisitionMonthDate'].isnull()]
```

Out[18]:

	AcquisitionID	AcquisitionMonth	AcquisitionMonthDate	AcquisitionYear	Company	
45	ACQ882	September	NaN	1997	Net Controls	
61	ACQ868	December	NaN	1998	Hyperparallel	D
99	ACQ833	NaN	NaN	2000	SoundJam MP[note 2]	
100	ACQ832	NaN	NaN	2001	Bluefish Labs	
144	ACQ793	February	NaN	2003	Pyra Labs	
149	ACQ789	April	NaN	2003	Applied Semantics	
150	ACQ788	April	NaN	2003	Neotonic Software	r
161	ACQ778	October	NaN	2003	Genius Labs	
162	ACQ777	October	NaN	2003	Sprinks	
166	ACQ773	January	NaN	2004	3721 Internet Assistant	H
182	ACQ759	September	NaN	2004	ZipDash	
184	ACQ757	October	NaN	2004	Where2	M
198	ACQ744	March	NaN	2005	Schemasoft	
205	ACQ738	April	NaN	2005	FingerWorks	
218	ACQ726	July	NaN	2005	Reqwireless	
233	ACQ712	November	NaN	2005	Skia Inc.	
301	ACQ651	December	NaN	2006	Wretch	
474	ACQ496	August	NaN	2010	Zetawire	
571	ACQ408	NaN	NaN	2012	WIMM Labs	s
629	ACQ356	NaN	NaN	2013	OttoCat	
630	ACQ355	NaN	NaN	2013	Novauris Technologies	
641	ACQ345	March	NaN	2013	osmeta	

	AcquisitionID	AcquisitionMonth	AcquisitionMonthDate	AcquisitionYear	Company	
713	ACQ280	December	NaN	2013	Acunu	
733	ACQ262	NaN	NaN	2014	Dryft	
840	ACQ166	January	NaN	2015	Camel Audio	in
858	ACQ15	October	NaN	2017	PowerbyProxi	
862	ACQ146	April	NaN	2015	Coherent Navigation	
869	ACQ14	October	NaN	2017	init.ai	
872	ACQ137	May	NaN	2015	Metaio	
890	ACQ120	September	NaN	2015	Perceptio	
892	ACQ119	September	NaN	2015	VocalIQ	
893	ACQ118	September	NaN	2015	Mapsense	
908	ACQ104	November	NaN	2015	Faceshift	

In [19]:

```
# Запоминаем индексы строк с пустыми значениями
flt_index = companies[companies['AcquisitionMonthDate'].isnull()].index
flt_index
```

Out[19]:

```
Int64Index([ 45,  61,  99, 100, 144, 149, 150, 161, 162, 166, 182, 184, 19
8,
            205, 218, 233, 301, 474, 571, 629, 630, 641, 713, 733, 840, 85
8,
            862, 869, 872, 890, 892, 893, 908],
           dtype='int64')
```

In [20]:

```
# Проверяем что выводятся нужные строки
companies[companies.index.isin(flt_index)]
```

Out[20]:

	AcquisitionID	AcquisitionMonth	AcquisitionMonthDate	AcquisitionYear	Company	
45	ACQ882	September	NaN	1997	Net Controls	
61	ACQ868	December	NaN	1998	Hyperparallel	D
99	ACQ833	NaN	NaN	2000	SoundJam MP[note 2]	
100	ACQ832	NaN	NaN	2001	Bluefish Labs	
144	ACQ793	February	NaN	2003	Pyra Labs	
149	ACQ789	April	NaN	2003	Applied Semantics	
150	ACQ788	April	NaN	2003	Neotonic Software	r
161	ACQ778	October	NaN	2003	Genius Labs	
162	ACQ777	October	NaN	2003	Sprinks	
166	ACQ773	January	NaN	2004	3721 Internet Assistant	H
182	ACQ759	September	NaN	2004	ZipDash	
184	ACQ757	October	NaN	2004	Where2	M
198	ACQ744	March	NaN	2005	Schemasoft	
205	ACQ738	April	NaN	2005	FingerWorks	
218	ACQ726	July	NaN	2005	Reqwireless	
233	ACQ712	November	NaN	2005	Skia Inc.	
301	ACQ651	December	NaN	2006	Wretch	
474	ACQ496	August	NaN	2010	Zetawire	
571	ACQ408	NaN	NaN	2012	WIMM Labs	sı
629	ACQ356	NaN	NaN	2013	OttoCat	
630	ACQ355	NaN	NaN	2013	Novauris Technologies	
641	ACQ345	March	NaN	2013	osmeta	

	AcquisitionID	AcquisitionMonth	AcquisitionMonthDate	AcquisitionYear	Company	
713	ACQ280	December	NaN	2013	Acunu	
733	ACQ262	NaN	NaN	2014	Dryft	
840	ACQ166	January	NaN	2015	Camel Audio	in
858	ACQ15	October	NaN	2017	PowerbyProxi	
862	ACQ146	April	NaN	2015	Coherent Navigation	
869	ACQ14	October	NaN	2017	init.ai	
872	ACQ137	May	NaN	2015	Metaio	
890	ACQ120	September	NaN	2015	Perceptio	
892	ACQ119	September	NaN	2015	VocalIQ	
893	ACQ118	September	NaN	2015	Mapsense	
908	ACQ104	November	NaN	2015	Faceshift	

In [21]:

```
# фильтр по колонке
data_num[data_num.index.isin(flt_index)]['AcquisitionMonthDate']
```

Out[21]:

```
45     NaN
61     NaN
99     NaN
100    NaN
144    NaN
149    NaN
150    NaN
161    NaN
162    NaN
166    NaN
182    NaN
184    NaN
198    NaN
205    NaN
218    NaN
233    NaN
301    NaN
474    NaN
571    NaN
629    NaN
630    NaN
641    NaN
713    NaN
733    NaN
840    NaN
858    NaN
862    NaN
869    NaN
872    NaN
890    NaN
892    NaN
893    NaN
908    NaN
Name: AcquisitionMonthDate, dtype: float64
```

In [22]:

```
#Будем использовать встроенные средства импьютации библиотеки scikit-learn - https://sc
ikit-learn.org/stable/modules/impute.html#impute
data_num_AcquisitionMonthDate = data_num[['AcquisitionMonthDate']]
data_num_AcquisitionMonthDate.head()
```

Out[22]:

	AcquisitionMonthDate
0	11.0
1	11.0
2	8.0
3	18.0
4	21.0

In [24]:

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

In [25]:

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

Out[25]:

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

```
[False],
[False],
[ True],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[ True],
[ True],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
```

```
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [ True],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
```

```
       [False],
       [ True],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
```

```
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
```

```
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
```

[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],

```
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
```

```
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
        [False],
```

```
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
```

```
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[ True],
[ True],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[ True],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
```

```
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
```

```
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
```

```
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[ True],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
```

```
[False],
[False],
[False],
[False],
[False],
[False],
[ True],
[False],
[False],
[False],
[ True],
[False],
[False],
[False],
[False],
[False],
[False],
[ True],
[False],
[False],
[ True],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[ True],
[False],
[ True],
[ True],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[False],
[ True],
[False],
[False],
[False],
[False],
```

```
       [False],
       [False],
       [False]])
```

In [26]:

```
#С помощью класса SimpleImputer можно проводить импьютацию различными показателями цент
ра распределения
strategies=['mean', 'median','most_frequent']
```

In [27]:

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

In [28]:

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

Out[28]:

```
('mean',
 array([14.70215176, 14.70215176, 14.70215176, 14.70215176, 14.70215176,
        14.70215176, 14.70215176, 14.70215176, 14.70215176, 14.70215176,
        14.70215176, 14.70215176, 14.70215176, 14.70215176, 14.70215176,
        14.70215176, 14.70215176, 14.70215176, 14.70215176, 14.70215176,
        14.70215176, 14.70215176, 14.70215176, 14.70215176, 14.70215176,
        14.70215176, 14.70215176, 14.70215176, 14.70215176, 14.70215176,
        14.70215176, 14.70215176, 14.70215176]))
```

In [29]:

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

Out[29]:

```
('median',
 array([14., 14., 14., 14., 14., 14., 14., 14., 14., 14., 14., 14., 14.,
        14., 14., 14., 14., 14., 14., 14., 14., 14., 14., 14., 14., 14.,
        14., 14., 14., 14., 14., 14., 14.]))
```

In [30]:

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

Out[30]:

```
('most_frequent',
 array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]))
```

In [31]:

```
# 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 [32]:

```
companies[['Value (USD)']].describe()
```

Out[32]:

	Value (USD)
count	2.450000e+02
mean	7.584170e+08
std	2.453624e+09
min	2.000000e+05
25%	3.000000e+07
50%	1.020000e+08
75%	4.500000e+08
max	2.620000e+10

In [33]:

```
test_num_impute_col(companies, 'Value (USD)', strategies[0])
```

Out[33]:

```
('Value (USD)', 'mean', 671, 758416979.5918367, 758416979.5918367)
```

In [34]:

```
test_num_impute_col(companies, 'Value (USD)', strategies[1])
```

Out[34]:

```
('Value (USD)', 'median', 671, 102000000.0, 102000000.0)
```

In [35]:

```
test_num_impute_col(companies, 'Value (USD)', strategies[2])
```

Out[35]:

```
('Value (USD)', 'most_frequent', 671, 100000000.0, 100000000.0)
```

In [36]:

```
#1.2.2. Обработка пропусков в категориальных данных
cars = pd.read_csv('Data/lab_3/Car_sales.csv', sep=',')
```

In [37]:

```
cars.isnull().sum()
```

Out[37]:

```
Manufacturer           0
Model                  0
Sales in thousands     0
4-year resale value    0
Vehicle type           0
Price in thousands     0
Engine size            0
Horsepower             0
Wheelbase              0
Width                  0
Length                 0
Curb weight            0
Fuel capacity          0
Fuel efficiency        0
Latest Launch          0
dtype: int64
```

Вывод: пропусков в данных нет, значит, они хорошо подходят для построения модели

In [38]:

```
companies2 = pd.read_csv('Data/lab_3/acquisitions.csv', sep=',')
companies2.head(5)
#companies2.shape
```

Out[38]:

	AcquisitionID	AcquisitionMonth	AcquisitionMonthDate	AcquisitionYear	Company	E
0	ACQ99	November	11.0	2015	bebop	Cloud
1	ACQ98	November	11.0	2015	Fly Labs	Vide
2	ACQ97	December	8.0	2015	Clearleap	Clou man
3	ACQ96	December	18.0	2015	Metanautix	
4	ACQ95	December	21.0	2015	Talko, Inc.	commu

In [39]:

```
#  возьмем старый датасет  companies
# Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета
cat_cols = []
for col in companies2.columns:
    # Количество пустых значений
    temp_null_count = companies2[companies2[col].isnull()].shape[0]
    dt = str(companies2[col].dtype)
    total_count = companies2.shape[0]
    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(c
ol, dt, temp_null_count, temp_perc))
```

```
Колонка AcquisitionMonth. Тип данных object. Количество пустых значений 6,
0.66%.
Колонка Country. Тип данных object. Количество пустых значений 46, 5.02%.
Колонка Derived products. Тип данных object. Количество пустых значений 51
5, 56.22%.
```

In [40]:

```
# Класс SimpleImputer можно использовать для категориальных признаков со стратегиями "m
ost_frequent" или "constant".
cat_temp_data = companies2[['Country']]
cat_temp_data.head(2)
```

Out[40]:

	Country
0	USA
1	USA

In [41]:

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

Out[41]:

```
array(['USA', 'GER', 'FIN', 'CAN', 'UK', 'SWE', 'ISR', 'TWN', 'AUS',
       'SGP', 'NOR', 'DEN', 'ROU', 'CHN', 'EU', 'IND', 'BLR', 'FRA',
       'BRA', 'ITA', 'SWI', 'SUI', 'CHE', 'NED', 'ESP', 'THA', 'BEL',
       'POR', nan, 'KOR', 'HKG', 'JOR', 'MYS', 'IRL', 'IDN', 'GRE', 'LUX',
       'UKR', 'AUT', 'JPN', 'NZL'], dtype=object)
```

In [42]:

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

Out[42]:

```
(46, 1)
```

In [43]:

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

Out[43]:

```
array([['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['GER'],
       ['FIN'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['CAN'],
       ['USA'],
       ['CAN'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['UK'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['GER'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['CAN'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['CAN'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
```

```
['GER'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['SWE'],
['CAN'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['GER'],
['USA'],
['USA'],
['GER'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['TWN'],
['USA'],
['USA'],
['USA'],
['AUS'],
['CAN'],
['USA'],
['USA'],
['CAN'],
['USA'],
['CAN'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
```

['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['SGP'],
['USA'],
['USA'],
['NOR'],
['GER'],
['DEN'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['AUS'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['ROU'],
['USA'],
['USA'],
['AUS'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CHN'],
['EU'],
['USA'],
['IND'],
['BLR'],
['FRA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],

```
['UK'],
['USA'],
['USA'],
['AUS'],
['EU'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['DEN'],
['USA'],
['USA'],
['FRA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['BRA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CHN'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['CAN'],
['BRA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['ITA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['SWI'],
['UK'],
['IND'],
['USA'],
['AUS'],
['USA'],
['USA'],
```

```
['USA'],
['GER'],
['GER'],
['USA'],
['USA'],
['ISR'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['FRA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['AUS'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['SUI'],
['USA'],
['USA'],
['ISR'],
['FRA'],
['USA'],
['GER'],
['BRA'],
['ISR'],
['USA'],
['USA'],
['AUS'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['EU'],
['TWN'],
['AUS'],
```

['USA'],
['CHE'],
['USA'],
['NED'],
['USA'],
['USA'],
['USA'],
['USA'],
['SWE'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['SWE'],
['SWI'],
['USA'],
['USA'],
['FRA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ESP'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['FIN'],
['USA'],
['THA'],
['DEN'],
['FRA'],
['USA'],
['CAN'],
['UK'],
['ISR'],
['USA'],
['USA'],

['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['ISR'],
['USA'],
['USA'],
['USA'],
['CAN'],
['SWE'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['NOR'],
['USA'],
['USA'],
['USA'],
['BEL'],
['USA'],
['USA'],
['POR'],
['USA'],
['USA'],
['USA'],
['FRA'],
['USA'],
['USA'],
['KOR'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['HKG'],
['ISR'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['JOR'],
['USA'],
['USA'],
['SGP'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],

```
['IND'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['MYS'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['IRL'],
['USA'],
['ISR'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['IDN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['IND'],
['USA'],
['USA'],
['CAN'],
['USA'],
```

['ISR'],
['UK'],
['USA'],
['SWE'],
['USA'],
['USA'],
['USA'],
['USA'],
['GRE'],
['ISR'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['AUS'],
['USA'],
['USA'],
['GER'],
['USA'],
['USA'],
['UK'],
['USA'],
['IRL'],
['ISR'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['LUX'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['NED'],
['USA'],
['USA'],
['FIN'],
['USA'],
['USA'],
['USA'],
['USA'],
['SWE'],
['USA'],
['USA'],
['USA'],
['UK'],
['CAN'],

```
['USA'],
['USA'],
['USA'],
['GER'],
['USA'],
['USA'],
['USA'],
['CAN'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['IRL'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['USA'],
['SWE'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['CAN'],
['USA'],
['ITA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['FRA'],
['SWE'],
['CAN'],
['USA'],
['USA'],
['USA'],
```

```
['USA'],
['USA'],
['USA'],
['ESP'],
['USA'],
['UK'],
['USA'],
['USA'],
['UKR'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['AUS'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['SWI'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['ISR'],
```

```
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['UK'],
['CHN'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['FIN'],
['USA'],
['SWE'],
['USA'],
['FIN'],
['USA'],
['USA'],
['UK'],
['USA'],
['IRL'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['AUT'],
['ISR'],
['FRA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['JPN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['GER'],
['USA'],
['USA'],
['USA'],
['USA'],
```

['USA'],
['USA'],
['USA'],
['USA'],
['CHE'],
['USA'],
['ISR'],
['USA'],
['USA'],
['IND'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['GER'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['UK'],
['UK'],
['FRA'],
['USA'],
['USA'],
['USA'],
['USA'],
['FIN'],
['FRA'],
['IND'],
['NZL'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],

```
['USA'],
['USA'],
['USA'],
['USA'],
['FRA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['FIN'],
['USA'],
['BLR'],
['USA'],
['ITA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['TWN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['IND'],
['NED'],
['USA'],
['CAN'],
['USA'],
['UK'],
['USA'],
['USA'],
['SWE'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['USA'],
['GER'],
['GER'],
['USA'],
['UK'],
['USA'],
['USA'],
['ISR'],
['IND'],
['UK'],
['USA'],
['FRA'],
['USA'],
['USA'],
['USA'],
['USA'],
```

```
['ISR'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['NZL'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['ISR'],
['USA'],
['USA'],
['IRL'],
['USA'],
['GER'],
['USA'],
['UK'],
['USA'],
['GER'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['ISR'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['AUS'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['IRL'],
['USA'],
['USA'],
['POR'],
['USA'],
['USA'],
['USA'],
['SWI'],
['USA'],
['USA'],
['USA'],
['ISR'],
```

```
       ['USA'],
       ['USA'],
       ['USA']], dtype=object)
```

In [44]:

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

Out[44]:

```
array(['AUS', 'AUT', 'BEL', 'BLR', 'BRA', 'CAN', 'CHE', 'CHN', 'DEN',
       'ESP', 'EU', 'FIN', 'FRA', 'GER', 'GRE', 'HKG', 'IDN', 'IND',
       'IRL', 'ISR', 'ITA', 'JOR', 'JPN', 'KOR', 'LUX', 'MYS', 'NED',
       'NOR', 'NZL', 'POR', 'ROU', 'SGP', 'SUI', 'SWE', 'SWI', 'THA',
       'TWN', 'UK', 'UKR', 'USA'], dtype=object)
```

In [45]:

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

Out[45]:

```
array([['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['GER'],
       ['FIN'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['CAN'],
       ['USA'],
       ['CAN'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['UK'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['GER'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['CAN'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['CAN'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
       ['USA'],
```

['GER'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['SWE'],
['CAN'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['GER'],
['USA'],
['USA'],
['GER'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['TWN'],
['USA'],
['USA'],
['USA'],
['AUS'],
['CAN'],
['USA'],
['USA'],
['CAN'],
['USA'],
['CAN'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],

['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['SGP'],
['USA'],
['USA'],
['NOR'],
['GER'],
['DEN'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['AUS'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['ROU'],
['USA'],
['USA'],
['AUS'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CHN'],
['EU'],
['USA'],
['IND'],
['BLR'],
['FRA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],

```
['UK'],
['USA'],
['USA'],
['AUS'],
['EU'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['DEN'],
['USA'],
['USA'],
['FRA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['BRA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CHN'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['CAN'],
['BRA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['ITA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['SWI'],
['UK'],
['IND'],
['USA'],
['AUS'],
['USA'],
['USA'],
```

```
['USA'],
['GER'],
['GER'],
['USA'],
['USA'],
['ISR'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['FRA'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['AUS'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['SUI'],
['USA'],
['USA'],
['ISR'],
['FRA'],
['USA'],
['GER'],
['BRA'],
['ISR'],
['USA'],
['USA'],
['AUS'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['EU'],
['TWN'],
['AUS'],
```

```
['USA'],
['CHE'],
['USA'],
['NED'],
['USA'],
['USA'],
['USA'],
['USA'],
['SWE'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['SWE'],
['SWI'],
['USA'],
['USA'],
['FRA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ESP'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['FIN'],
['USA'],
['THA'],
['DEN'],
['FRA'],
['USA'],
['CAN'],
['UK'],
['ISR'],
['USA'],
['USA'],
```

```
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['ISR'],
['USA'],
['USA'],
['USA'],
['CAN'],
['SWE'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['NOR'],
['USA'],
['USA'],
['USA'],
['BEL'],
['USA'],
['USA'],
['POR'],
['USA'],
['USA'],
['!!!'],
['FRA'],
['USA'],
['USA'],
['KOR'],
['USA'],
['USA'],
['USA'],
['!!!'],
['USA'],
['HKG'],
['ISR'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['JOR'],
['USA'],
['USA'],
['SGP'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
```

```
['IND'],
['USA'],
['USA'],
['CAN'],
['USA'],
['!!!'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['MYS'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['!!!'],
['UK'],
['USA'],
['IRL'],
['!!!'],
['ISR'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['!!!'],
['USA'],
['USA'],
['USA'],
['IDN'],
['USA'],
['USA'],
['!!!'],
['USA'],
['USA'],
['USA'],
['!!!'],
['USA'],
['CAN'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['IND'],
['USA'],
['USA'],
['CAN'],
['USA'],
```

```
['ISR'],
['UK'],
['USA'],
['SWE'],
['USA'],
['USA'],
['USA'],
['USA'],
['GRE'],
['ISR'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['UK'],
['!!!'],
['USA'],
['AUS'],
['USA'],
['USA'],
['GER'],
['USA'],
['USA'],
['UK'],
['USA'],
['IRL'],
['ISR'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['LUX'],
['!!!'],
['USA'],
['!!!'],
['!!!'],
['CAN'],
['USA'],
['USA'],
['USA'],
['NED'],
['USA'],
['!!!'],
['FIN'],
['USA'],
['USA'],
['USA'],
['USA'],
['SWE'],
['USA'],
['!!!'],
['USA'],
['UK'],
['CAN'],
```

```
['USA'],
['USA'],
['USA'],
['GER'],
['!!!'],
['USA'],
['USA'],
['CAN'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['!!!'],
['USA'],
['IRL'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['!!!'],
['!!!'],
['ISR'],
['USA'],
['USA'],
['!!!'],
['USA'],
['SWE'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['!!!'],
['USA'],
['USA'],
['USA'],
['!!!'],
['UK'],
['CAN'],
['USA'],
['ITA'],
['USA'],
['!!!'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['FRA'],
['SWE'],
['CAN'],
['USA'],
['USA'],
['!!!'],
```

```
['USA'],
['USA'],
['USA'],
['ESP'],
['USA'],
['UK'],
['USA'],
['USA'],
['UKR'],
['USA'],
['!!!'],
['USA'],
['!!!'],
['USA'],
['USA'],
['AUS'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['!!!'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['SWI'],
['USA'],
['USA'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['!!!'],
['USA'],
['USA'],
['USA'],
['USA'],
['!!!'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['ISR'],
```

```
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['UK'],
['CHN'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['!!!'],
['ISR'],
['USA'],
['USA'],
['FIN'],
['USA'],
['SWE'],
['!!!'],
['FIN'],
['!!!'],
['USA'],
['UK'],
['USA'],
['IRL'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['AUT'],
['ISR'],
['FRA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['JPN'],
['USA'],
['!!!'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['GER'],
['USA'],
['USA'],
['USA'],
['USA'],
```

['USA'],
['USA'],
['USA'],
['USA'],
['CHE'],
['USA'],
['ISR'],
['USA'],
['USA'],
['IND'],
['USA'],
['USA'],
['USA'],
['USA'],
['!!!'],
['UK'],
['USA'],
['GER'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['USA'],
['UK'],
['UK'],
['FRA'],
['!!!'],
['USA'],
['USA'],
['!!!'],
['FIN'],
['FRA'],
['IND'],
['NZL'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['!!!'],
['!!!'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['!!!'],
['USA'],
['USA'],

```
['USA'],
['USA'],
['USA'],
['USA'],
['FRA'],
['USA'],
['ISR'],
['USA'],
['USA'],
['FIN'],
['!!!'],
['BLR'],
['USA'],
['ITA'],
['CAN'],
['USA'],
['!!!'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['TWN'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['IND'],
['NED'],
['USA'],
['CAN'],
['USA'],
['UK'],
['USA'],
['!!!'],
['SWE'],
['USA'],
['ISR'],
['USA'],
['USA'],
['USA'],
['USA'],
['GER'],
['GER'],
['USA'],
['UK'],
['USA'],
['USA'],
['ISR'],
['IND'],
['UK'],
['USA'],
['FRA'],
['USA'],
['USA'],
['USA'],
['!!!'],
```

```
['ISR'],
['USA'],
['USA'],
['USA'],
['!!!'],
['USA'],
['NZL'],
['USA'],
['USA'],
['USA'],
['USA'],
['!!!'],
['USA'],
['USA'],
['CAN'],
['ISR'],
['USA'],
['USA'],
['IRL'],
['!!!'],
['GER'],
['USA'],
['UK'],
['USA'],
['GER'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['ISR'],
['ISR'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['AUS'],
['UK'],
['USA'],
['USA'],
['USA'],
['USA'],
['USA'],
['CAN'],
['USA'],
['USA'],
['IRL'],
['USA'],
['USA'],
['POR'],
['USA'],
['!!!'],
['USA'],
['SWI'],
['USA'],
['USA'],
['USA'],
['ISR'],
```

```
       ['USA'],
       ['USA'],
       ['USA']], dtype=object)
```

In [46]:

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

Out[46]:

```
array(['!!!', 'AUS', 'AUT', 'BEL', 'BLR', 'BRA', 'CAN', 'CHE', 'CHN',
       'DEN', 'ESP', 'EU', 'FIN', 'FRA', 'GER', 'GRE', 'HKG', 'IDN',
       'IND', 'IRL', 'ISR', 'ITA', 'JOR', 'JPN', 'KOR', 'LUX', 'MYS',
       'NED', 'NOR', 'NZL', 'POR', 'ROU', 'SGP', 'SUI', 'SWE', 'SWI',
       'THA', 'TWN', 'UK', 'UKR', 'USA'], dtype=object)
```

In [47]:

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

Out[47]:

```
46
```

In [48]:

```
#2. Преобразование категориальных признаков
cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
cat_enc
```

Out[48]:

	c1
0	USA
1	USA
2	USA
3	USA
4	USA
5	USA
6	GER
7	FIN
8	USA
9	USA
10	USA
11	USA
12	USA
13	USA
14	USA
15	CAN
16	USA
17	CAN
18	USA
19	USA
20	USA
21	USA
22	UK
23	USA
24	USA
25	USA
26	USA
27	USA
28	USA
29	GER
...	...
886	USA
887	USA
888	USA
889	USA
890	USA
891	AUS

	c1
892	UK
893	USA
894	USA
895	USA
896	USA
897	USA
898	CAN
899	USA
900	USA
901	IRL
902	USA
903	USA
904	POR
905	USA
906	USA
907	USA
908	SWI
909	USA
910	USA
911	USA
912	ISR
913	USA
914	USA
915	USA

916 rows × 1 columns

In [49]:

```
# 2.1. Кодирование категорий целочисленными значениями - label encoding
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [50]:

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

In [51]:

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

Out[51]:

```
array(['USA', 'GER', 'FIN', 'CAN', 'UK', 'SWE', 'ISR', 'TWN', 'AUS',
       'SGP', 'NOR', 'DEN', 'ROU', 'CHN', 'EU', 'IND', 'BLR', 'FRA',
       'BRA', 'ITA', 'SWI', 'SUI', 'CHE', 'NED', 'ESP', 'THA', 'BEL',
       'POR', 'KOR', 'HKG', 'JOR', 'MYS', 'IRL', 'IDN', 'GRE', 'LUX',
       'UKR', 'AUT', 'JPN', 'NZL'], dtype=object)
```

In [52]:

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

Out[52]:

```
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])
```

In [53]:

```
le.inverse_transform([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])
```

Out[53]:

```
array(['AUS', 'AUT', 'BEL', 'BLR', 'BRA', 'CAN', 'CHE', 'CHN', 'DEN',
       'ESP', 'EU', 'FIN', 'FRA', 'GER', 'GRE', 'HKG', 'IDN', 'IND',
       'IRL', 'ISR', 'ITA', 'JOR', 'JPN', 'KOR', 'LUX', 'MYS', 'NED',
       'NOR', 'NZL', 'POR', 'ROU', 'SGP', 'SUI', 'SWE', 'SWI', 'THA',
       'TWN', 'UK', 'UKR', 'USA'], dtype=object)
```

In [54]:

```
# можно вывести часть значений
le.inverse_transform([0, 1, 2, 3, 4, 5])
```

Out[54]:

```
array(['AUS', 'AUT', 'BEL', 'BLR', 'BRA', 'CAN'], dtype=object)
```

In [55]:

```
# 2.2. Кодирование категорий наборами бинарных значений - one-hot encoding
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])
cat_enc.shape
```

Out[55]:

```
(916, 1)
```

In [56]:

```
cat_enc_ohe.shape
```

Out[56]:

```
(916, 40)
```

In [57]:

```
cat_enc_ohe
```

Out[57]:

```
<916x40 sparse matrix of type '<class 'numpy.float64'>'
	with 916 stored elements in Compressed Sparse Row format>
```

In [58]:

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

Out[58]:

```
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., 0., 0., 0., 1.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 1.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 1.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 1.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 1.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 1.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 1.],
        [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., 0., 0., 0., 1.]])
```

In [59]:

```
cat_enc.head(10)
```

Out[59]:

	c1
0	USA
1	USA
2	USA
3	USA
4	USA
5	USA
6	GER
7	FIN
8	USA
9	USA

In [60]:

```
# 2.3. Pandas get_dummies - быстрый вариант one-hot кодирования
pd.get_dummies(cat_enc).head(10)
# единицы проставляются там, где совпадение значения
```

Out[60]:

	c1_AUS	c1_AUT	c1_BEL	c1_BLR	c1_BRA	c1_CAN	c1_CHE	c1_CHN	c1_DEN	c1_
0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	

10 rows × 40 columns

In [61]:

```
pd.get_dummies(cat_temp_data, dummy_na=True).head()
```

Out[61]:

	Country_AUS	Country_AUT	Country_BEL	Country_BLR	Country_BRA	Country_CAN	Cc
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 × 41 columns

In [62]:

```
# попробуем для другого датасета
cat_temp_data2 = companies2[['ParentCompany']]
```

In [63]:

```
pd.get_dummies(cat_temp_data2, dummy_na=True).head(8)
```

Out[63]:

	ParentCompany_Apple	ParentCompany_Facebook	ParentCompany_Google	ParentCompa
0	0	0	1	
1	0	0	1	
2	0	0	0	
3	0	0	0	
4	0	0	0	
5	1	0	0	
6	0	0	0	
7	0	0	0	

In [64]:

```
 # 3. Масштабирование данных
    # Термины "масштабирование" и "нормализация" часто используются как синонимы. Масшт
абирование предполагает изменение диапазона измерения величины, а нормализация - измене
ние распределения этой величины.
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
    # 3.1. MinMax масштабирование
```

In [65]:

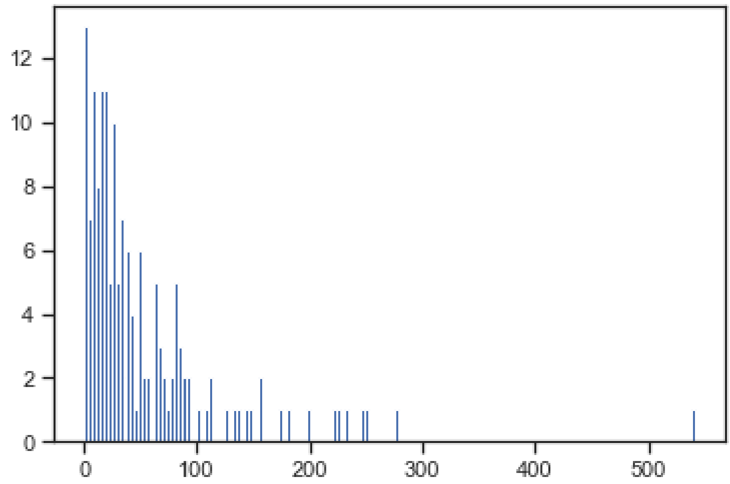
```
#возьмем датасет car_sales
cars.head()
cars.shape
```

Out[65]:

```
(157, 15)
```

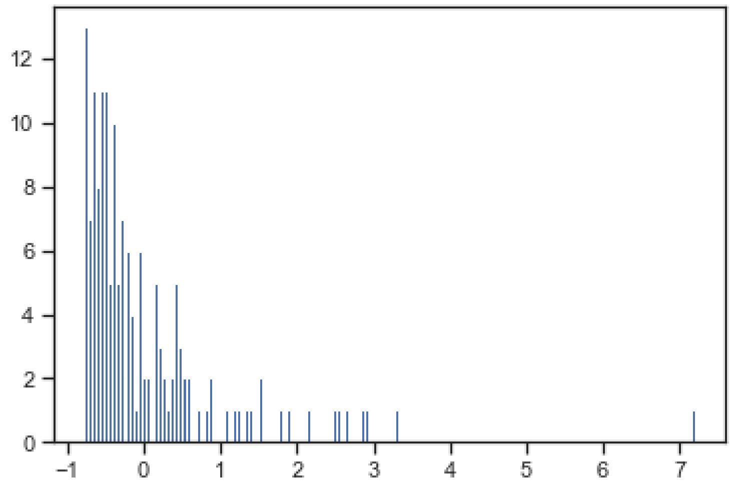
In [66]:

```
sc2 = StandardScaler()
#cars.dtypes
sc2_data = sc2.fit_transform(cars[['Sales in thousands']])
plt.hist(cars['Sales in thousands'], 157)
plt.show()
```



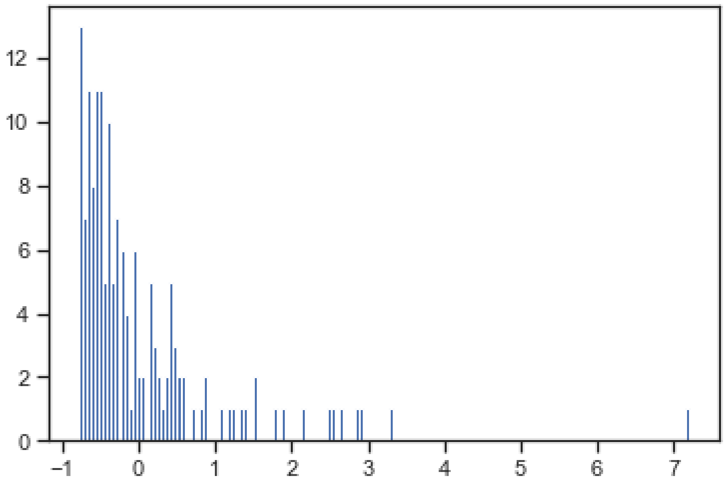
In [67]:

```
plt.hist(sc2_data, 157)
plt.show()
```



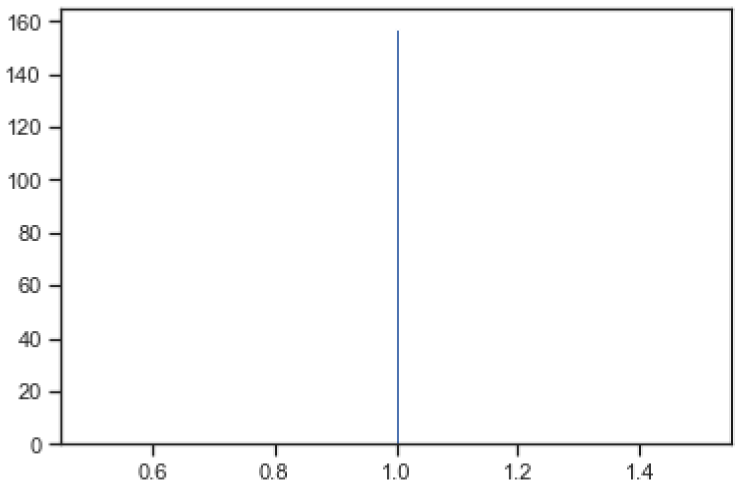
In [68]:

```
#3.2. Масштабирование данных на основе Z-оценки - StandardScaler
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(cars[['Sales in thousands']])
plt.hist(sc2_data, 157)
plt.show()
# Масштабирование на основе z-оценки похоже на масштабирование MinMax
```



In [69]:

```
# 3.3. Нормализация данных
sc3 = Normalizer()
sc3_data = sc3.fit_transform(cars[['Sales in thousands']])
plt.hist(sc3_data, 157)
plt.show()
```



In [ ]:

```
```