Московский Государственный технический университет имени Н. Э. Баумана

Лабораторная работа No2 по курсу: «Технология машинного обучения»

Работу выполнил студент группы ИУ5-63

Федорова Антонина_____

Работу проверил:

Гапанюк Ю.Е._____

Москва 2019

Задание:

1. 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.)
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:

- обработку пропусков в данных;
- кодирование категориальных признаков;
- масштабирование данных.

Текст программы с примерами выполнения программы:

Обработка пропусков в данных

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

```
drive.mount("/content/gdrive", force_remount=True)
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?cli

Enter your authorization code:
··········
Mounted at /content/gdrive
```

```
data = pd.read_csv('/content/gdrive/My Drive/countries of the world.csv', sep=",
```

```
data.head()
```

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Ne migratio
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48,0	0,00	23,0
1	Albania	EASTERN EUROPE	3581655	28748	124,6	1,26	-4,9
2	Algeria	NORTHERN AFRICA	32930091	2381740	13,8	0,04	-0,3
3	American Samoa	OCEANIA	57794	199	290,4	58,29	-20,7
4	Andorra	WESTERN EUROPE	71201	468	152,1	0,00	6,

```
data.shape
```

```
(227, 20)
```

```
data.dtypes
```

```
Country                               object
Region                                object
Population                             int64
Area (sq. mi.)                         int64
Pop. Density (per sq. mi.)            object
Coastline (coast/area ratio)          object
Net migration                         object
Infant mortality (per 1000 births)    object
GDP ($ per capita)                   float64
Literacy (%)                          object
Phones (per 1000)                     object
Arable (%)                            object
Crops (%)                             object
Other (%)                             object
Climate                               object
Birthrate                             object
Deathrate                             object
Agriculture                           object
Industry                              object
Service                               object
dtype: object
```

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

```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         3
Infant mortality (per 1000 births)    3
GDP ($ per capita)                    1
Literacy (%)                         18
Phones (per 1000)                     4
Arable (%)                            2
Crops (%)                             2
Other (%)                             2
Climate                              22
Birthrate                             3
Deathrate                             4
Agriculture                          15
Industry                             16
Service                              15
dtype: int64
```

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

```
((227, 20), (227, 6))
```

Видно, что удалять колонки в данном случае- не выход. Из 20 колонок осталось только 6, а это уже очень плохо.

```
data_str = data.dropna(axis=0, how='any')
print(data.shape, data_str.shape)
round(data_str.shape[0]/data.shape[0], 2)
```

```
(227, 20) (179, 20)
0.79
```

Видно, что таким способом мы потеряли больше 20% датасета. В моем случае такие потери данных критичны так как я взяла небольшой датасет.

```
num_cols = []
for col in data.columns:
    null_count = data[data[col].isnull()].shape[0]
    col_type = str(data[col].dtype)
    if null_count>0 and (col_type=='float64' or col_type=='int64'):
        num_cols.append(col)
        temp_perc = round((null_count / data.shape[0]) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.f
```

```
Колонка GDP ($ per capita). Тип данных float64. Количество пустых значений
```

Получается, в моем датасете только одна колонка с нулевым значением в числовой фиче. Причем, нулевое значение только одно.

```
null_index = data[data['GDP ($ per capita)'].isnull()].index
null_index
```

```
Int64Index([223], dtype='int64')
```

```
data[data.index.isin(null_index)]
```

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	mo (p
3	Western Sahara	NORTHERN AFRICA	273008	266000	1,0	0,42	NaN	

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

```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data[['GDP ($ per capita)']])
```

```
imp_num = SimpleImputer(strategy='mean')
data_num_imp = imp_num.fit_transform(data[['GDP ($ per capita)']])
data_num_imp[mask_missing_values_only]
```

```
array([9689.82300885])
```

Теперь надо разобраться с пропусками в категориальных фичах. Я выберу те колонки, в которых меньше всего пропущенных значений.

```
num_cols = []
for col in data.columns:
    null_count = data[data[col].isnull()].shape[0]
    col_type = str(data[col].dtype)
    if null_count>0 and (col_type=='object'):
        num_cols.append(col)
        temp_perc = round((null_count / data.shape[0]) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.f
```

```
Колонка Net migration. Тип данных object. Количество пустых значений 3, 1.
Колонка Infant mortality (per 1000 births). Тип данных object. Количество
Колонка Literacy (%). Тип данных object. Количество пустых значений 18, 7.
Колонка Phones (per 1000). Тип данных object. Количество пустых значений 4
Колонка Arable (%). Тип данных object. Количество пустых значений 2, 0.88%
Колонка Crops (%). Тип данных object. Количество пустых значений 2, 0.88%.
Колонка Other (%). Тип данных object. Количество пустых значений 2, 0.88%.
Колонка Climate. Тип данных object. Количество пустых значений 22, 9.69%.
Колонка Birthrate. Тип данных object. Количество пустых значений 3, 1.32%.
Колонка Deathrate. Тип данных object. Количество пустых значений 4, 1.76%.
Колонка Agriculture. Тип данных object. Количество пустых значений 15, 6.6
Колонка Industry. Тип данных object. Количество пустых значений 16, 7.05%.
Колонка Service. Тип данных object. Количество пустых значений 15, 6.61%.
```

```
cat_temp_data = data[['Crops (%)']]
cat_temp_data.head()
```

	Crops (%)
0	0,22
1	4,42
2	0,25
3	15
4	0

```
cat_temp_data['Crops (%)'].unique()
```

```
array(['0,22', '4,42', '0,25', '15', '0', '0,24', '4,55', '0,48', '2,3',
       '0,04', '0,86', '2,71', '0,4', '5,63', '3,07', '2,33', '0,6',
       '1,71', '2,4', '0,43', '0,19', '2,96', '0,01', '0,9', '6,67',
       '0,76', '1,92', '0,97', '14,02', '0,61', '2,58', '0,02', '0,5',
       '0,14', '0,42', '1,25', '1,67', '23,32', '0,52', '0,13', '13,04',
       '5,88', '13,84', '2,27', '7,6', '4,44', '3,05', '20', '10,33',
       '0,67', '4,93', '12,07', '3,57', '0,03', '0,45', '0,75', '4,65',
       '2,07', '0,05', '5,46', '0,66', '21,05', '3,86', '0,59', '9,67',
       '8,78', '29,41', '3,55', '16,36', '5,03', nan, '8,82', '0,15',
       '11,61', '3,22', '1,01', '2,06', '2,74', '7,23', '1,39', '0,78',
       '4,17', '9,53', '10,16', '0,96', '1,83', '0,98', '50,68', '2,49',
       '1,95', '0,11', '0,35', '0,47', '13,98', '2,28', '0,91', '1,81',
       '1,03', '1,49', '17,61', '16,67', '3,13', '38,89', '9,43', '1,31',
       '45,71', '10,79', '2,17', '0,3', '0,64', '0,33', '6,99', '1,94',
       '4,35', '0,87', '1,98', '1,44', '0,23', '16,77', '1,12', '7,81',
       '5,52', '0,27', '1,2', '2,25', '12,16', '2,78', '22,95', '17,95',
       '24,38', '48,96', '0,09', '0,21', '3,2', '13,33', '0,89', '2,62',
       '2', '0,79', '9,87', '15,7', '0,18', '0,06', '0,7', '4,43', '1',
       '0,92', '1,08', '6,46', '2,21', '43,06', '9,16', '13,74', '3,31',
       '10,65', '1,61', '0,83', '7,38', '5,95', '2,94', '25', '18,97',
       '0,34'], dtype=object)
```

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

```
       ['0,11'],
       ['12,16'],
       ['0'],
       ['2,78'],
       ['22,95'],
       ['0'],
       ['17,95'],
       ['24,38'],
       ['0'],
       ['48,96'],
       ['0,09'],
       ['0,21'],
       ['3,2'],
       ['13,33'],
       ['0,89'],
       ['0'],
       ['2,62'],
       ['1,49'],
       ['2'],
       ['0,04'],
       ['0,79'],
       ['9,87'],
       ['15,7'],
```

```
['0,18'],
['0,06'],
['0,7'],
['0,01'],
['0,61'],
['4,43'],
['1'],
['0,92'],
['1,08'],
['6,46'],
['2,21'],
['43,06'],
['9,16'],
['13,74'],
['3,31'],
['0,14'],
['0'],
['0'],
['10,65'],
['1,61'],
['2,25'],
['0,21'],
['0,22'],
['0,23'],
['0,83'],
['7,38'],
['0,92'],
['5,95'],
['2,94'],
['25'],
['18,97'],
['0'],
['0,24'],
['0,03'],
['0,34']], dtype=object)
```

```
np.unique(data_imp2)
```

```
array(['0', '0,01', '0,02', '0,03', '0,04', '0,05', '0,06', '0,09',
       '0,11', '0,13', '0,14', '0,15', '0,18', '0,19', '0,21', '0,22',
       '0,23', '0,24', '0,25', '0,27', '0,3', '0,33', '0,34', '0,35',
       '0,4', '0,42', '0,43', '0,45', '0,47', '0,48', '0,5', '0,52',
       '0,59', '0,6', '0,61', '0,64', '0,66', '0,67', '0,7', '0,75',
       '0,76', '0,78', '0,79', '0,83', '0,86', '0,87', '0,89', '0,9',
       '0,91', '0,92', '0,96', '0,97', '0,98', '1', '1,01', '1,03',
       '1,08', '1,12', '1,2', '1,25', '1,31', '1,39', '1,44', '1,49',
       '1,61', '1,67', '1,71', '1,81', '1,83', '1,92', '1,94', '1,95',
       '1,98', '10,16', '10,33', '10,65', '10,79', '11,61', '12,07',
       '12,16', '13,04', '13,33', '13,74', '13,84', '13,98', '14,02',
       '15', '15,7', '16,36', '16,67', '16,77', '17,61', '17,95', '18,97',
       '2', '2,06', '2,07', '2,17', '2,21', '2,25', '2,27', '2,28', '2,3',
       '2,33', '2,4', '2,49', '2,58', '2,62', '2,71', '2,74', '2,78',
       '2,94', '2,96', '20', '21,05', '22,95', '23,32', '24,38', '25',
       '29,41', '3,05', '3,07', '3,13', '3,2', '3,22', '3,31', '3,55',
       '3,57', '3,86', '38,89', '4,17', '4,35', '4,42', '4,43', '4,44',
       '4,55', '4,65', '4,93', '43,06', '45,71', '48,96', '5,03', '5,46',
       '5,52', '5,63', '5,88', '5,95', '50,68', '6,46', '6,67', '6,99',
       '7,23', '7,38', '7,6', '7,81', '8,78', '8,82', '9,16', '9,43',
       '9,53', '9,67', '9,87'], dtype=object)
```

```
# Импьютация константой
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='<3'
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

```
array([['0,22'],
       ['4,42'],
       ['0,25'],
       ['15'],
       ['0'],
       ['0,24'],
       ['0'],
       ['4,55'],
       ['0,48'],
       ['2,3'],
       ['0'],
       ['0,04'],
       ['0,86'],
       ['2,71'],
       ['0,4'],
       ['5,63'],
       ['3,07'],
       ['2,33'],
       ['0,6'],
       ['0,4'],
       ['1,71'],
       ['2,4'],
       ['0'],
```

```
['0,43'],
['0,19'],
['2,96'],
['0,01'],
['0,9'],
['6,67'],
['0,76'],
['1,92'],
['0,19'],
['0,97'],
['14,02'],
['0,61'],
['2,58'],
['0,02'],
['0,5'],
['0'],
['0,14'],
['0,02'],
['0,42'],
['1,25'],
['1,67'],
['23,32'],
['0,52'],
['0,13'],
['13,04'],
['5,88'],
['13,84'],
['2,27'],
['7,6'],
['4,44'],
['3,05'],
['0,19'],
['0'],
['20'],
['10,33'],
['0,67'],
```

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

```
array(['0', '0,01', '0,02', '0,03', '0,04', '0,05', '0,06', '0,09',
       '0,11', '0,13', '0,14', '0,15', '0,18', '0,19', '0,21', '0,22',
       '0,23', '0,24', '0,25', '0,27', '0,3', '0,33', '0,34', '0,35',
       '0,4', '0,42', '0,43', '0,45', '0,47', '0,48', '0,5', '0,52',
       '0,59', '0,6', '0,61', '0,64', '0,66', '0,67', '0,7', '0,75',
       '0,76', '0,78', '0,79', '0,83', '0,86', '0,87', '0,89', '0,9',
       '0,91', '0,92', '0,96', '0,97', '0,98', '1', '1,01', '1,03',
       '1,08', '1,12', '1,2', '1,25', '1,31', '1,39', '1,44', '1,49',
       '1,61', '1,67', '1,71', '1,81', '1,83', '1,92', '1,94', '1,95',
       '1,98', '10,16', '10,33', '10,65', '10,79', '11,61', '12,07',
       '12,16', '13,04', '13,33', '13,74', '13,84', '13,98', '14,02',
       '15', '15,7', '16,36', '16,67', '16,77', '17,61', '17,95', '18,97',
       '2', '2,06', '2,07', '2,17', '2,21', '2,25', '2,27', '2,28', '2,3',
       '2,33', '2,4', '2,49', '2,58', '2,62', '2,71', '2,74', '2,78',
       '2,94', '2,96', '20', '21,05', '22,95', '23,32', '24,38', '25',
       '29,41', '3,05', '3,07', '3,13', '3,2', '3,22', '3,31', '3,55',
       '3,57', '3,86', '38,89', '4,17', '4,35', '4,42', '4,43', '4,44',
       '4,55', '4,65', '4,93', '43,06', '45,71', '48,96', '5,03', '5,46',
       '5,52', '5,63', '5,88', '5,95', '50,68', '6,46', '6,67', '6,99',
       '7,23', '7,38', '7,6', '7,81', '8,78', '8,82', '9,16', '9,43',
       '9,53', '9,67', '9,87', '<3'], dtype=object)
```

Преобразование категориальных признаков в числовые

Для данной задачи я выбрала другой датасет.

```
data2 = pd.read_csv('/content/gdrive/My Drive/googleplaystore.csv', sep=",")
```

```
data2.head(10)
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0

	Mouna							
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free	0
6	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178	19M	50,000+	Free	0
7	Infinite Painter	ART_AND_DESIGN	4.1	36815	29M	1,000,000+	Free	0
8	Garden Coloring Book	ART_AND_DESIGN	4.4	13791	33M	1,000,000+	Free	0
9	Kids Paint Free - Drawing Fun	ART_AND_DESIGN	4.7	121	3.1M	10,000+	Free	0

Так как это уже другой датасет, необходимо проверить его на нулевые значения.

```
data2.isnull().sum()
```

```
App                0
Category           0
Rating          1474
Reviews            0
Size               0
Installs           0
Type               1
Price              0
Content Rating     1
Genres             0
Last Updated       0
Current Ver        8
Android Ver        3
dtype: int64
```

В пункте "Category" нет ни одного нулевого значения. Именно поэтому далее я буду рассматривать его. В этой колонке описывается категория, к которой относится приложение.

```
data2.shape
```

```
(10841, 13)
```

```
cat_data = data2[['Category']]
```

```
print(data2['Category'].unique().size)
data2['Category'].unique()
```

```
34
array(['ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FINANCE',
       'FOOD_AND_DRINK', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME',
       'LIBRARIES_AND_DEMO', 'LIFESTYLE', 'GAME', 'FAMILY', 'MEDICAL',
       'SOCIAL', 'SHOPPING', 'PHOTOGRAPHY', 'SPORTS', 'TRAVEL_AND_LOCAL',
       'TOOLS', 'PERSONALIZATION', 'PRODUCTIVITY', 'PARENTING', 'WEATHER',
       'VIDEO_PLAYERS', 'NEWS_AND_MAGAZINES', 'MAPS_AND_NAVIGATION',
       '1.9'], dtype=object)
```

Видно, что всего уникальных значений у этого признака - 34. Следовательно, этот признак можно **закодировать целочисленными значениями**.

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
le = LabelEncoder()
cat_enc_le = le.fit_transform(data2[['Category']])
```

```
r/local/lib/python3.6/dist-packages/sklearn/preprocessing/label.py:235: Dat
  = column_or_1d(y, warn=True)
```

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

```
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]
```

```
le.inverse_transform([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14
      17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33])
```

```
array(['1.9', 'ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FAMILY',
       'FINANCE', 'FOOD_AND_DRINK', 'GAME', 'HEALTH_AND_FITNESS',
       'HOUSE_AND_HOME', 'LIBRARIES_AND_DEMO', 'LIFESTYLE',
       'MAPS_AND_NAVIGATION', 'MEDICAL', 'NEWS_AND_MAGAZINES',
       'PARENTING', 'PERSONALIZATION', 'PHOTOGRAPHY', 'PRODUCTIVITY',
       'SHOPPING', 'SOCIAL', 'SPORTS', 'TOOLS', 'TRAVEL_AND_LOCAL',
       'VIDEO_PLAYERS', 'WEATHER'], dtype=object)
```

```
print(data2['Type'].unique().size)
data2['Type'].unique()
```

```
4
array(['Free', 'Paid', nan, '0'], dtype=object)
```

Так как в типе есть одно пропущенное значение, я просто удалю строку, в которой оно содержится.

```
data2 = data2.dropna(axis=0, how='any')
data2.shape
```

```
(9360, 13)
```

```
print(data2['Type'].unique().size)
data2['Type'].unique()
```

```
2
array(['Free', 'Paid'], dtype=object)
```

Теперь в данной колонке осталось только 2 значения: бесплатное и платное.

```
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(data2[['Type']])
```

```
data2[['Type']].shape
```

```
(9360, 1)
```

```
cat_enc_ohe.shape
```

```
(9360, 2)
```

```
cat_enc_ohe.todense()[7030:7040]
```

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

```
data2[['Type']][7030:7040]
```

	Type
7895	Free
7896	Free
7899	Paid
7900	Free
7902	Free
7903	Free
7904	Free
7905	Free
7906	Free
7907	Free

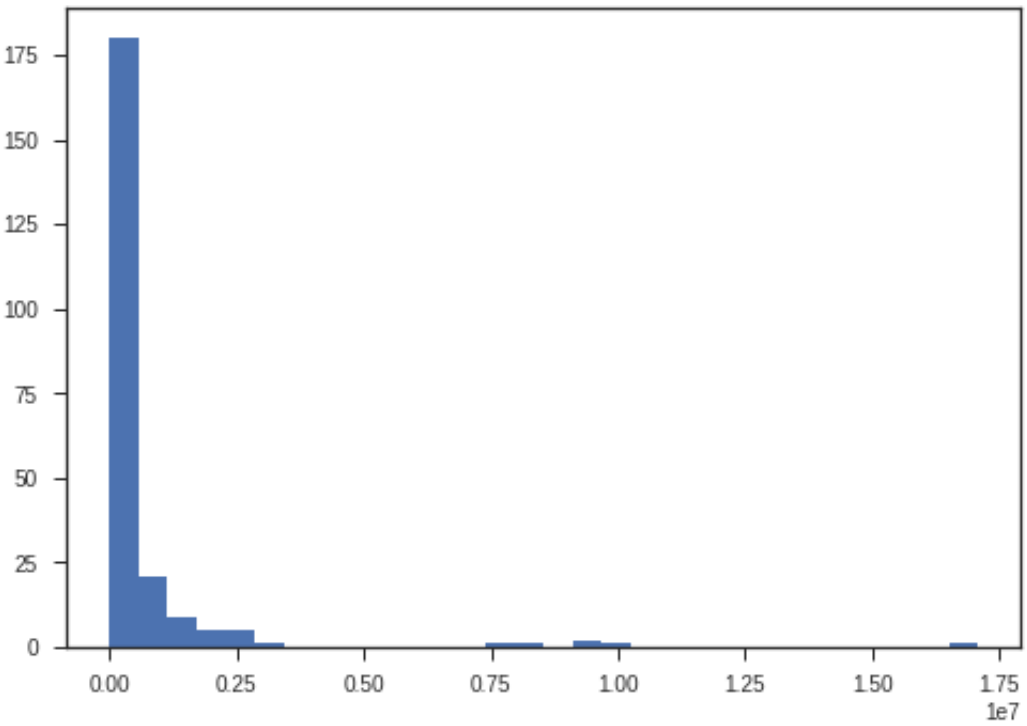
```
pd.get_dummies(data2[['Type']])[7030:7040]
```

	Type_Free	Type_Paid
7895	1	0
7896	1	0
7899	0	1
7900	1	0
7902	1	0
7903	1	0
7904	1	0
7905	1	0
7906	1	0
7907	1	0

Масштабирование данных

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

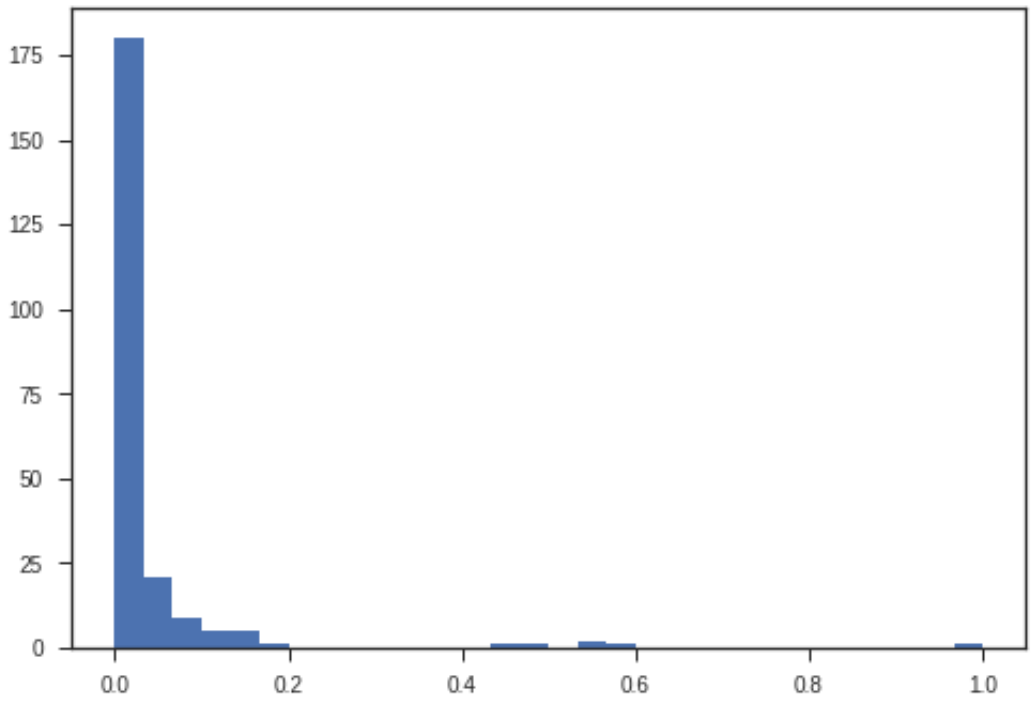
```
plt.hist(data['Area (sq. mi.)'], 30)
plt.show()
```




```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['Area (sq. mi.)']])
```

```
/local/lib/python3.6/dist-packages/sklearn/preprocessing/data.py:334: DataC
turn self.partial_fit(X, y)
```

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

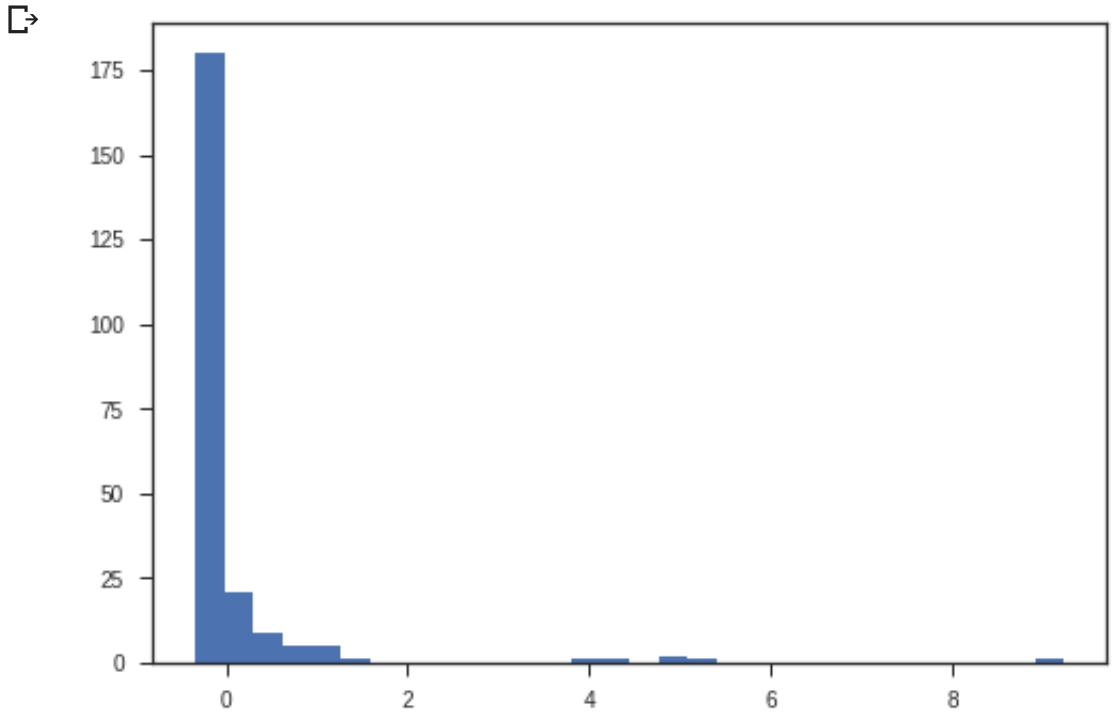


Масштабирование данных на основе Z-оценки - StandardScaler

```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Area (sq. mi.)']])
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/data.py:645:
  return self.partial_fit(X, y)
/usr/local/lib/python3.6/dist-packages/sklearn/base.py:464: DataConversion
  return self.fit(X, **fit_params).transform(X)
```

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



Нормализация данных

```
sc3 = Normalizer()
sc3_data = sc3.fit_transform(data[['Area (sq. mi.)']])
```

```
plt.hist(sc3_data, 10)
plt.show()
```