Лабораторная работа №2

April 15, 2021

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

1.1 1. Описание задания.

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.

1.2 2. Выполнение работы.

Перед началом работы подключаем необходимые библиотеки:

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

1.2.1 Обработка пропусков данных.

Для обработки пропуска заднных выберем подходящий датасет.

[3]:
```
data = pd.read_csv('country_vaccinations.csv', sep=",")
```

Данный датасет содержит данные о прогрессе вакцинации в разных странах.

[66]:
```
data.shape
```

[66]:
```
(11156, 15)
```

Проверим, есть ли пропуски, которые мы могли бы устранить:

[18]:
```
data.isnull().sum()
```

[18]:
```
country                                0
iso_code                               0
date                                   0
total_vaccinations                  4510
people_vaccinated                   5169
people_fully_vaccinated             6872
daily_vaccinations_raw              5590
daily_vaccinations                   196
total_vaccinations_per_hundred      4510
people_vaccinated_per_hundred       5169
people_fully_vaccinated_per_hundred 6872
daily_vaccinations_per_million       196
vaccines                               0
source_name                            0
source_website                         0
dtype: int64
```

[22]:
```
data.head()
```

[22]:
```
       country iso_code        date  total_vaccinations ⊠
 ↪people_vaccinated  \
0  Afghanistan      AFG  2021-02-22                 0.0              ⊠
 ↪ 0.0
1  Afghanistan      AFG  2021-02-23                 NaN              ⊠
 ↪ NaN
2  Afghanistan      AFG  2021-02-24                 NaN              ⊠
 ↪ NaN
3  Afghanistan      AFG  2021-02-25                 NaN              ⊠
 ↪ NaN
4  Afghanistan      AFG  2021-02-26                 NaN              ⊠
 ↪ NaN

   people_fully_vaccinated  daily_vaccinations_raw ⊠
 ↪daily_vaccinations  \
0                      NaN                     NaN                NaN
1                      NaN                     NaN             1367.0
2                      NaN                     NaN             1367.0
3                      NaN                     NaN             1367.0
4                      NaN                     NaN             1367.0

   total_vaccinations_per_hundred  people_vaccinated_per_hundred  \
```

```
0                                0.0                                0.0
1                                NaN                                NaN
2                                NaN                                NaN
3                                NaN                                NaN
4                                NaN                                NaN

   people_fully_vaccinated_per_hundred ☒
 ↪daily_vaccinations_per_million  \
0                                    NaN                                NaN
1                                    NaN                               35.0
2                                    NaN                               35.0
3                                    NaN                               35.0
4                                    NaN                               35.0

             vaccines                 source_name  \
0  Oxford/AstraZeneca  Government of Afghanistan
1  Oxford/AstraZeneca  Government of Afghanistan
2  Oxford/AstraZeneca  Government of Afghanistan
3  Oxford/AstraZeneca  Government of Afghanistan
4  Oxford/AstraZeneca  Government of Afghanistan

                                       source_website
0  http://www.xinhuanet.com/english/asiapacific/2...
1  http://www.xinhuanet.com/english/asiapacific/2...
2  http://www.xinhuanet.com/english/asiapacific/2...
3  http://www.xinhuanet.com/english/asiapacific/2...
4  http://www.xinhuanet.com/english/asiapacific/2...
```

Судя по полученным сведениям, датасет имеет множество пропусков, которые необходимо устранить.

Удаление и заполнение нулями.

[23]:
```
# Удаление колонок, содержащих пустые значения
data_del_1 = data.dropna(axis=1, how='any')
(data.shape, data_del_1.shape)
```

[23]:
```
((11156, 15), (11156, 6))
```

[24]:
```
# Удаление строк, содержащих пустые значения
data_del_2 = data.dropna(axis=0, how='any')
(data.shape, data_del_2.shape)
```

[24]:
```
((11156, 15), (3837, 15))
```

[25]:
```
# Заполнение всех пропущенных значений нулями
data_new_3 = data.fillna(0)
data_new_3.head()
```

```
[25]:       country iso_code        date  total_vaccinations
    ↪people_vaccinated  \
0  Afghanistan      AFG  2021-02-22                 0.0
    ↪ 0.0
1  Afghanistan      AFG  2021-02-23                 0.0
    ↪ 0.0
2  Afghanistan      AFG  2021-02-24                 0.0
    ↪ 0.0
3  Afghanistan      AFG  2021-02-25                 0.0
    ↪ 0.0
4  Afghanistan      AFG  2021-02-26                 0.0
    ↪ 0.0

   people_fully_vaccinated  daily_vaccinations_raw
    ↪daily_vaccinations  \
0                      0.0                     0.0                 0.0
1                      0.0                     0.0              1367.0
2                      0.0                     0.0              1367.0
3                      0.0                     0.0              1367.0
4                      0.0                     0.0              1367.0

   total_vaccinations_per_hundred  people_vaccinated_per_hundred  \
0                             0.0                            0.0
1                             0.0                            0.0
2                             0.0                            0.0
3                             0.0                            0.0
4                             0.0                            0.0

   people_fully_vaccinated_per_hundred
    ↪daily_vaccinations_per_million  \
0                                  0.0                          0.0
1                                  0.0                         35.0
2                                  0.0                         35.0
3                                  0.0                         35.0
4                                  0.0                         35.0

             vaccines                source_name  \
0  Oxford/AstraZeneca  Government of Afghanistan
1  Oxford/AstraZeneca  Government of Afghanistan
2  Oxford/AstraZeneca  Government of Afghanistan
3  Oxford/AstraZeneca  Government of Afghanistan
4  Oxford/AstraZeneca  Government of Afghanistan

                                      source_website
0  http://www.xinhuanet.com/english/asiapacific/2…
1  http://www.xinhuanet.com/english/asiapacific/2…
```

```
2  http://www.xinhuanet.com/english/asiapacific/2...
3  http://www.xinhuanet.com/english/asiapacific/2...
4  http://www.xinhuanet.com/english/asiapacific/2...
```

Внедрение значений - импьютация Все колонки числовые, поэтому просто составим их список и определим количество пустых значений:

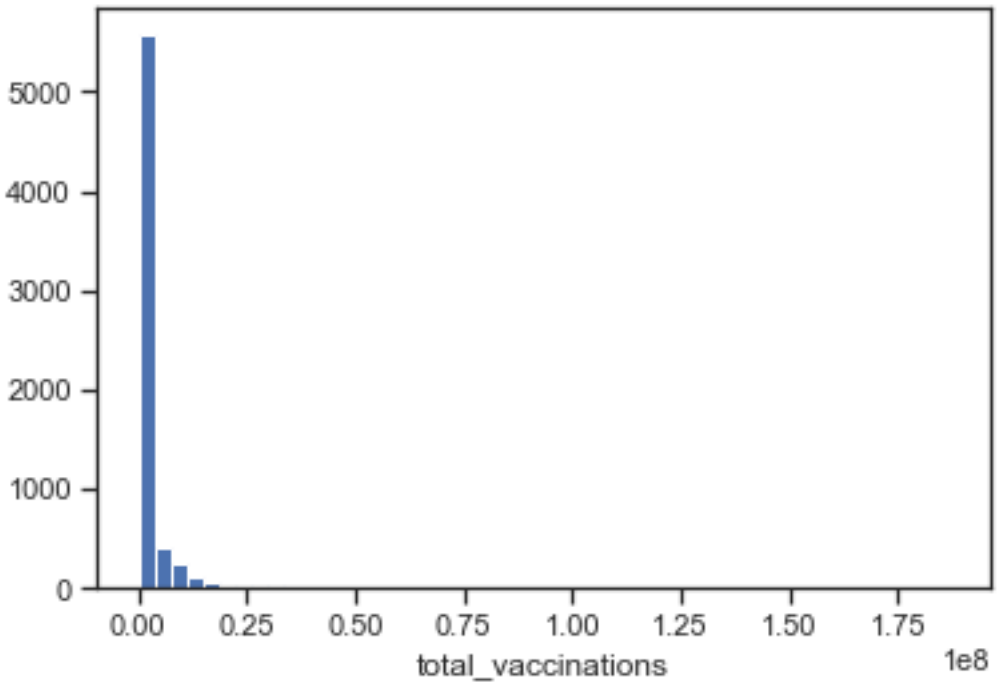
[29]:
```
nul_cols = ['total_vaccinations', 'people_vaccinated',
 ↪'people_fully_vaccinated', 'daily_vaccinations_raw',
 ↪'daily_vaccinations', 'total_vaccinations_per_hundred',
 ↪'people_vaccinated_per_hundred']
total_count = data.shape[0]
for col in nul_cols:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0:
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений
 ↪{}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

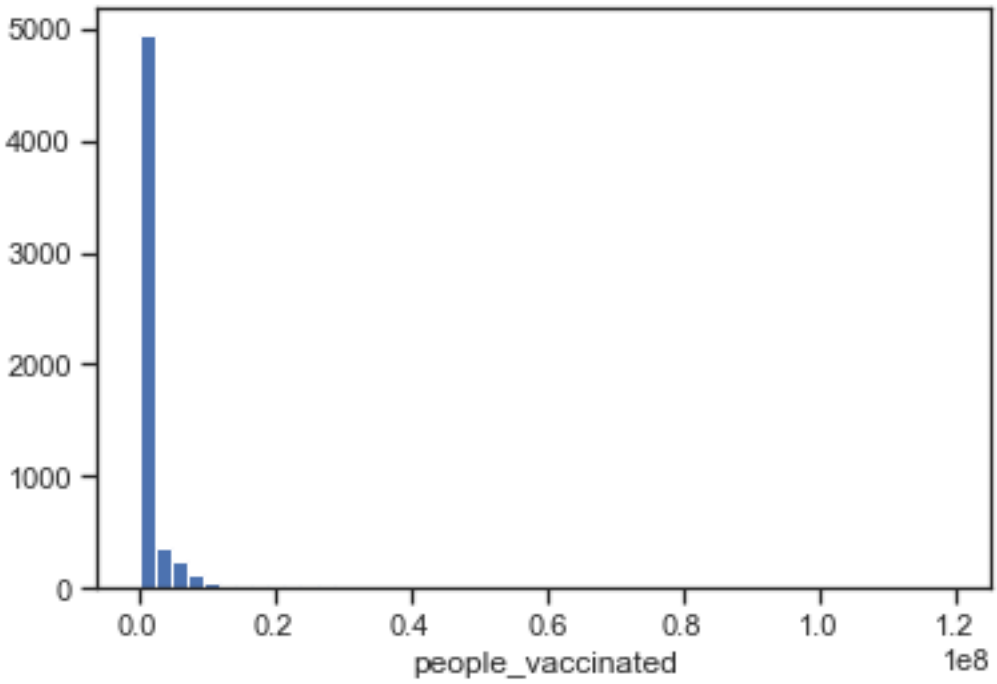
```
Колонка total_vaccinations. Тип данных float64. Количество пустых
 ↪значений 4510,
40.43%.
Колонка people_vaccinated. Тип данных float64. Количество пустых
 ↪значений 5169,
46.33%.
Колонка people_fully_vaccinated. Тип данных float64. Количество пустых
 ↪значений
6872, 61.6%.
Колонка daily_vaccinations_raw. Тип данных float64. Количество пустых
 ↪значений
5590, 50.11%.
Колонка daily_vaccinations. Тип данных float64. Количество пустых
 ↪значений 196,
1.76%.
Колонка total_vaccinations_per_hundred. Тип данных float64. Количество
 ↪пустых
значений 4510, 40.43%.
Колонка people_vaccinated_per_hundred. Тип данных float64. Количество
 ↪пустых
значений 5169, 46.33%.
```

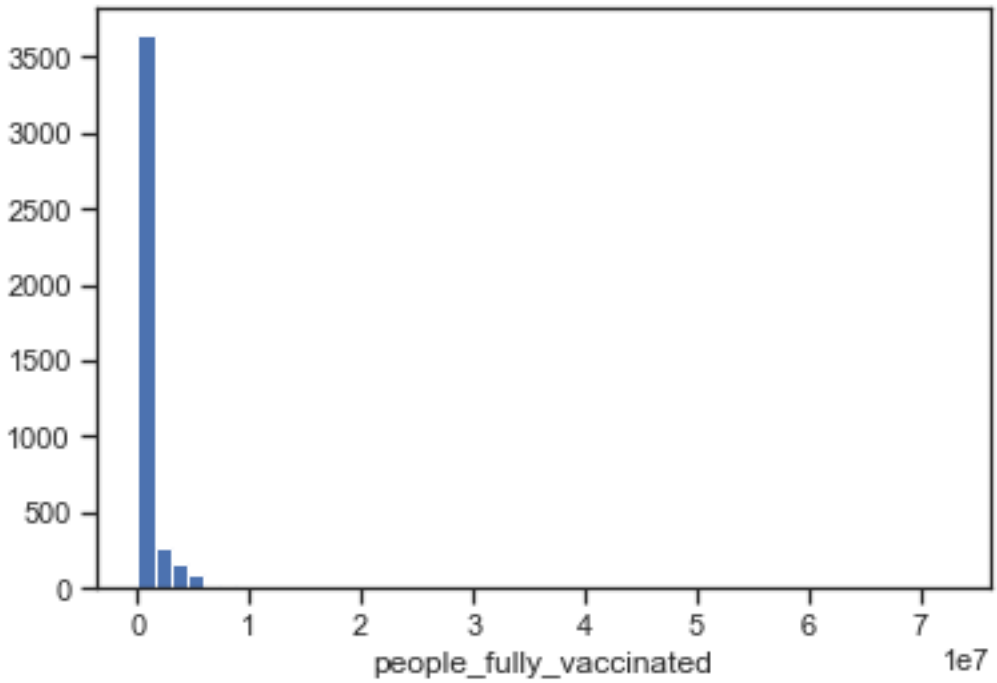
[32]:
```
# Фильтр по колонкам с пропущенными значениями
data_nul = data[nul_cols]
data_nul
```

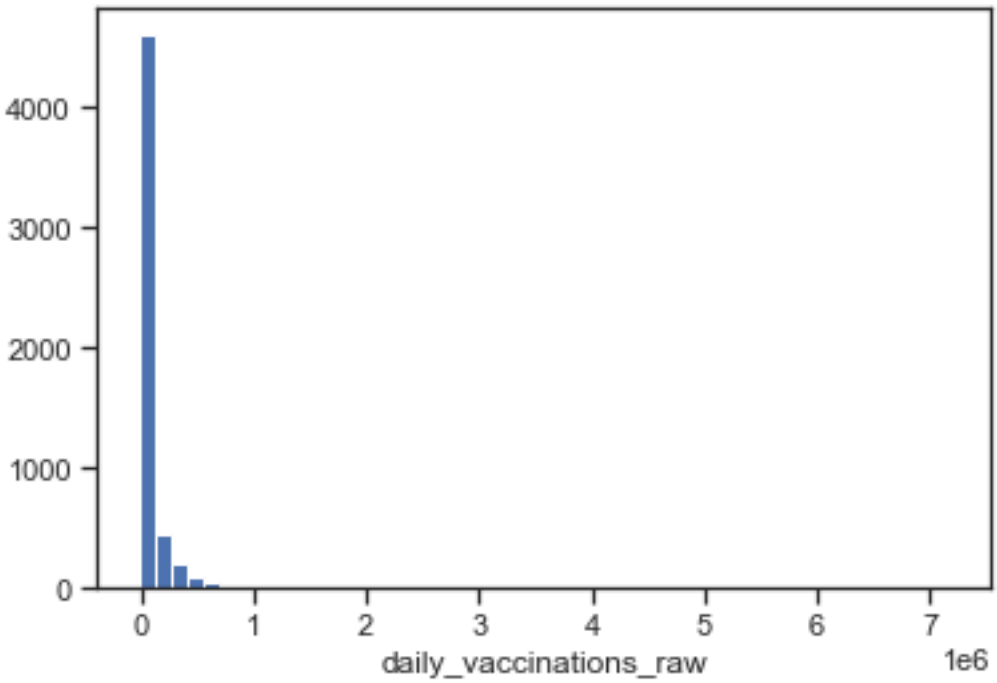
[32]:
```
       total_vaccinations  people_vaccinated  people_fully_vaccinated
  ↪ \
0                     0.0                0.0                      NaN
1                     NaN                NaN                      NaN
2                     NaN                NaN                      NaN
3                     NaN                NaN                      NaN
4                     NaN                NaN                      NaN
…                     …                  …                        …
11151            162633.0           139133.0                  23500.0
11152            179417.0           153238.0                  26179.0
11153            193677.0           166543.0                  27134.0
11154            206205.0           178237.0                  27968.0
11155            222733.0           193936.0                  28797.0

       daily_vaccinations_raw  daily_vaccinations  \
0                         NaN                 NaN
1                         NaN              1367.0
2                         NaN              1367.0
3                         NaN              1367.0
4                         NaN              1367.0
…                         …                …
11151                 17123.0             10967.0
11152                 16784.0             12505.0
11153                 14260.0             12624.0
11154                 12528.0             11636.0
11155                 16528.0             12831.0

       total_vaccinations_per_hundred  people_vaccinated_per_hundred
0                                0.00                           0.00
1                                 NaN                            NaN
2                                 NaN                            NaN
3                                 NaN                            NaN
4                                 NaN                            NaN
…                                 …                              …
11151                            1.09                           0.94
11152                            1.21                           1.03
11153                            1.30                           1.12
11154                            1.39                           1.20
11155                            1.50                           1.30

[11156 rows x 7 columns]
```

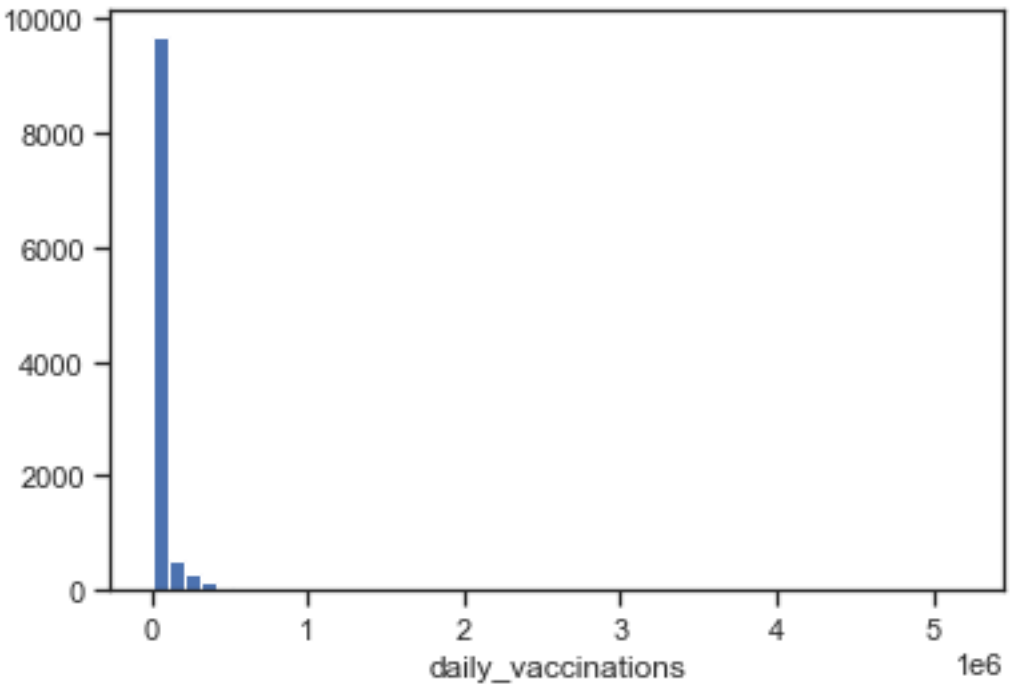
[33]:
```
# Гистограмма по признакам
for col in data_nul:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```

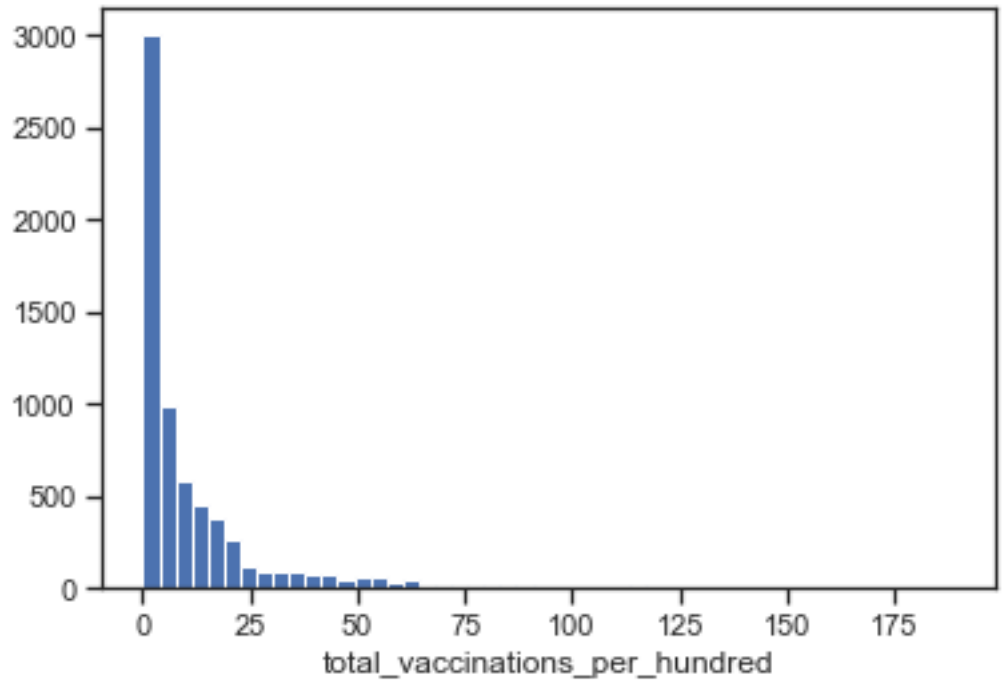

5000
4000
3000
2000
1000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
total_vaccinations
1e8

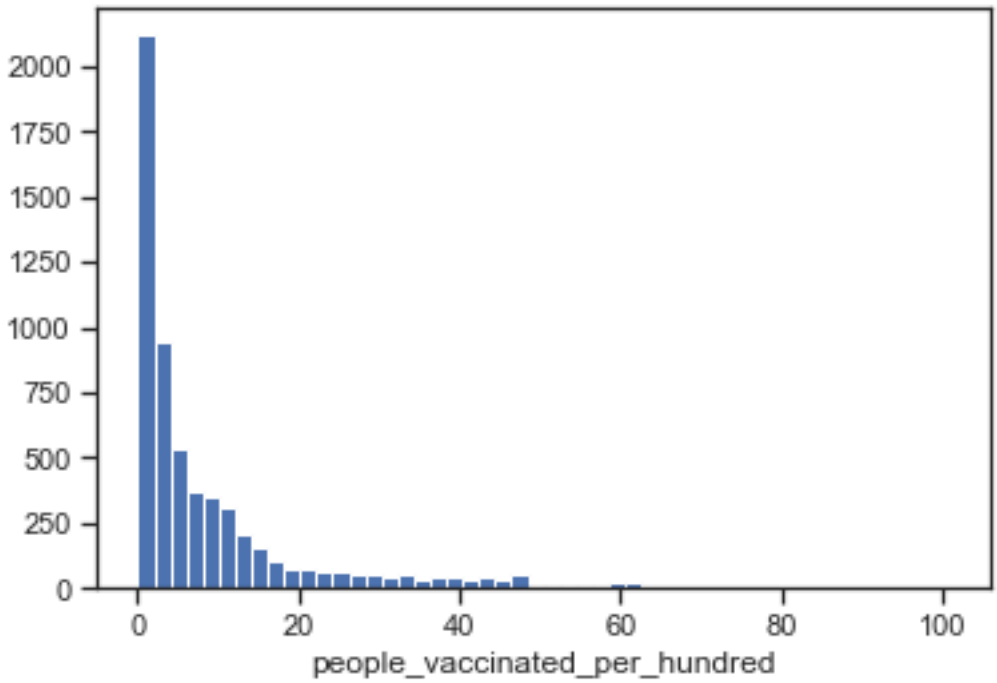

5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
people_vaccinated
1e8


3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
7
people_fully_vaccinated
1e7


4000
3000
2000
1000
0
0
1
2
3
4
5
6
7
daily_vaccinations_raw
1e6


10000
8000
6000
4000
2000
0
0
1
2
3
4
5
daily_vaccinations
1e6


3000
2500
2000
1500
1000
500
0
0
25
50
75
100
125
150
175
total_vaccinations_per_hundred



Можно использовать встроенные средства импьютации библиотеки scikit-learn:

[37]:
```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

Выберем один из столбцов:

[43]:
```
data_nul_PeopleVaccinatedPerHundred =
 ↪data_nul[['people_vaccinated_per_hundred']]
data_nul_PeopleVaccinatedPerHundred.head()
```

[43]:
```
   people_vaccinated_per_hundred
0                            0.0
1                            NaN
2                            NaN
3                            NaN
4                            NaN
```

[45]:
```
# Фильтр для проверки заполнения пустых значений для одного из
 ↪столбцов
indicator = MissingIndicator()
mask_missing_values_only = indicator.
 ↪fit_transform(data_nul_PeopleVaccinatedPerHundred)
mask_missing_values_only
```

[45]:
```
array([[False],
       [ True],
       [ True],
       …,
       [False],
       [False],
       [False]])
```

Далее используем функцию SimpleImputer с использованием различных показателей центра распределения (среднее значение, медиана, наиболее часто встречающееся значение).

[61]:
```
strategies=['mean', 'median', 'most_frequent']
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_nul_imp = imp_num.
 ↪fit_transform(data_nul_PeopleVaccinatedPerHundred)
    return data_nul_imp[mask_missing_values_only]
```

[62]:
```
strategies[0], test_num_impute(strategies[0])
```

[62]:
```
('mean',
 array([9.35741607, 9.35741607, 9.35741607, …, 9.35741607, 9.35741607,
        9.35741607]))
```

[63]:
```
strategies[1], test_num_impute(strategies[1])
```

[63]:
```
('median', array([3.81, 3.81, 3.81, …, 3.81, 3.81, 3.81]))
```

[65]:
```
strategies[2], test_num_impute(strategies[2])
```

[65]:
```
('most_frequent', array([0., 0., 0., …, 0., 0., 0.]))
```

1.2.2 Кодирование категориальных признаков.

Для исправления пропусков в категориальных признаках выберем другой датасет, так как в предыдущем все пропуски в числовых столбцах.

Тема данного датасета - тв шоу и сериалы сервиса Netflix. Подключим данный датасет.

[67]:
```
data2 = pd.read_csv('netflix_titles.csv', sep=",")
data2.isnull().sum()
```

[67]:
```
show_id          0
type             0
title            0
director      2389
cast           718
country        507
```

```
date_added        10
release_year       0
rating             7
duration           0
listed_in          0
description        0
dtype: int64
```

Выделим один столбец, где удалим пропуски. Пусть это будет столбец author.

[68]:
```
cat_enc = pd.DataFrame({'c1':data2['director']})
cat_enc
```

[68]:
```
                    c1
0                  NaN
1     Jorge Michel Grau
2          Gilbert Chan
3           Shane Acker
4        Robert Luketic
…                  …
7782        Josef Fares
7783        Mozez Singh
7784                NaN
7785                NaN
7786           Sam Dunn

[7787 rows x 1 columns]
```

Кодирование категорий целочисленными значениями. Импортируем все необходимые библиотеки sklearn. Используем метод label encoder.

[69]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

[70]:
```
le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
```

Просмотрим список имеющихся уникальных значений:

[71]:
```
cat_enc['c1'].unique()
```

[71]:
```
array([nan, 'Jorge Michel Grau', 'Gilbert Chan', …, 'Josef Fares',
       'Mozez Singh', 'Sam Dunn'], dtype=object)
```

[72]:
```
np.unique(cat_enc_le)
```

[72]:
```
array([   0,    1,    2, …, 4047, 4048, 4049])
```

[73]:
```
le.inverse_transform([0])
```

[73]:
```
array(['A. L. Vijay'], dtype=object)
```

Попробуем метод one-hot encoding.

[74]:
```
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])
```

[75]:
```
cat_enc.shape
```

[75]:
```
(7787, 1)
```

[76]:
```
cat_enc_ohe.shape
```

[76]:
```
(7787, 4050)
```

[77]:
```
cat_enc_ohe.todense()[0:10]
```

[77]:
```
matrix([[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.]])
```

[78]:
```
pd.get_dummies(cat_enc).head(10)
```

[78]:

	c1_A. L. Vijay	c1_A. Raajdheep	c1_A. Salaam	c1_A.R. Murugadoss	c1_Aadish Keluskar	c1_Aamir Bashir	c1_Aamir Khan	c1_Aanand Rai
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0				
7	0	0	0	0				
8	0	0	0	0				
9	0	0	0	0				

```
6                    0                    0                    0                    0
7                    0                    0                    0                    0
8                    0                    0                    0                    0
9                    0                    0                    0                    0

   c1_Aaron Burns  c1_Aaron Hancox, Michael McNamara  …  \
0               0                                   0  …
1               0                                   0  …
2               0                                   0  …
3               0                                   0  …
4               0                                   0  …
5               0                                   0  …
6               0                                   0  …
7               0                                   0  …
8               0                                   0  …
9               0                                   0  …

   c1_Álex de la Iglesia  c1_Álvaro Brechner  c1_Álvaro
 ↪Delgado-Aparicio L.  \
0                     0                    0                     
 ↪    0
1                     0                    0                     
 ↪    0
2                     0                    0                     
 ↪    0
3                     0                    0                     
 ↪    0
4                     0                    0                     
 ↪    0
5                     0                    0                     
 ↪    0
6                     0                    0                     
 ↪    0
7                     0                    0                     
 ↪    0
8                     0                    0                     
 ↪    0
9                     0                    0                     
 ↪    0

   c1_Álvaro Longoria, Gerardo Olivares  c1_Ángel Gómez Hernández  \
0                                     0                          0
1                                     0                          0
2                                     0                          0
3                                     0                          0
4                                     0                          0
```

```
5                                    0                                    0
6                                    0                                    0
7                                    0                                    0
8                                    0                                    0
9                                    0                                    0

   c1_Çagan Irmak  c1_Ísold Uggadóttir  c1_Óskar Thór Axelsson  \
0               0                     0                       0
1               0                     0                       0
2               0                     0                       0
3               0                     0                       0
4               0                     0                       0
5               0                     0                       0
6               0                     0                       0
7               0                     0                       0
8               0                     0                       0
9               0                     0                       0

   c1_Ömer Faruk Sorak  c1_Şenol Sönmez
0                    0                0
1                    0                0
2                    0                0
3                    0                0
4                    0                0
5                    0                0
6                    0                0
7                    0                0
8                    0                0
9                    0                0

[10 rows x 4049 columns]
```

1.2.3 Масштабирование данных.

Данная операция неоюходима для того, чтобы привести диапазоны величин, имеющих пропуски, к примерно одной области. Подберем датасет, подходящий для этой операции.

[83]:
```
data3 = pd.read_csv('oasis_cross-sectional.csv', sep=",")
data3.describe()
```

[83]:
```
                Age         Educ          SES        MMSE          CDR      ☒
  ↪    eTIV  \
count  436.000000  235.000000  216.000000  235.00000  235.000000  ☒
  ↪436.000000
mean    51.357798    3.178723    2.490741   27.06383    0.285106 ☒
  ↪1481.919725
```

```
std      25.269862    1.311510    1.120593     3.69687    0.383405
 ↪158.740866
min      18.000000    1.000000    1.000000    14.00000    0.000000
 ↪1123.000000
25%      23.000000    2.000000    2.000000    26.00000    0.000000
 ↪1367.750000
50%      54.000000    3.000000    2.000000    29.00000    0.000000
 ↪1475.500000
75%      74.000000    4.000000    3.000000    30.00000    0.500000
 ↪1579.250000
max      96.000000    5.000000    5.000000    30.00000    2.000000
 ↪1992.000000
```

	nWBV	ASF	Delay
count	436.000000	436.000000	20.00000
mean	0.791670	1.198894	20.55000
std	0.059937	0.128682	23.86249
min	0.644000	0.881000	1.00000
25%	0.742750	1.111750	2.75000
50%	0.809000	1.190000	11.00000
75%	0.842000	1.284250	30.75000
max	0.893000	1.563000	89.00000

Пропуски содержатся в поле eTIV и там же можно нормализовать данные сместив их к диапазону от нуля до единицы.

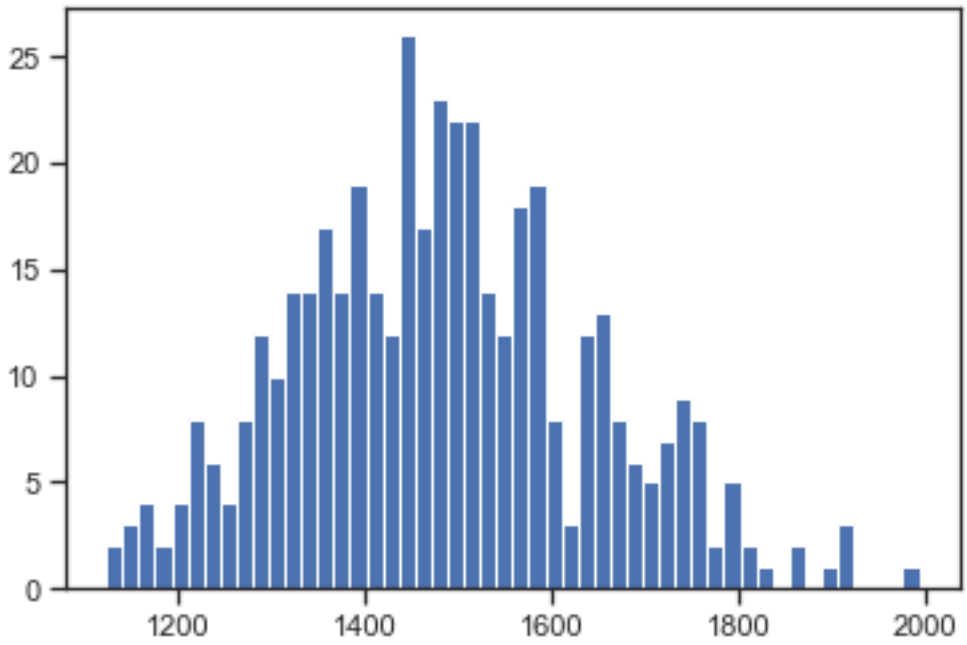
Импортируем необходимые библиотеки. Импортируем библиотеки.

[88]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler,
 ↪Normalizer
```

[90]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data3[['eTIV']])
```

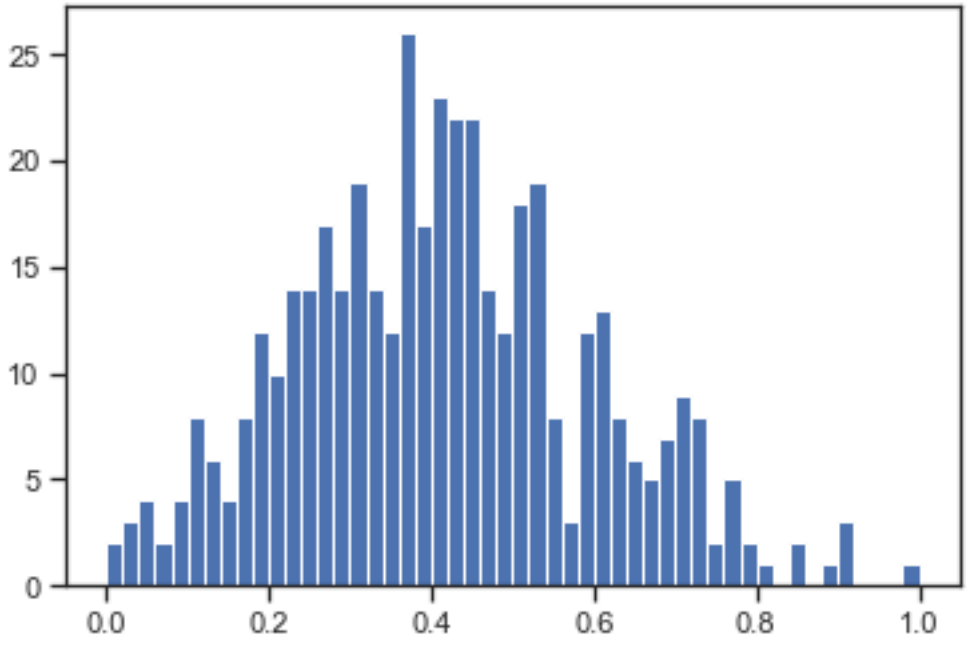
Рассмотрим сходный график.

[93]:
```
plt.hist(data3['eTIV'], 50)
plt.show()
```



И рассмотрим полученный график.

[92]:
```
plt.hist(sc1_data, 50)
plt.show()
```



Также можно использовать масштабирование данных на основе Z-оценки, позволяющее сместить данные в диапазон от 0 до 3. Проверим этот метод на том же признаке.

[95]:
```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data3[['eTIV']])
```

[96]:
```
plt.hist(sc2_data, 50)
plt.show()
```