Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №2
«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных.»

Выполнил:
студент группы РТ5-61Б
Агеев А
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2023 г

Описание задания

- Выбрать набор данных (датасет).
- Для первой лабораторной работы рекомендуется использовать датасет без пропусков в данных, например из Scikit-learn.

Для лабораторных работ не рекомендуется выбирать датасеты большого размера.

- Создать ноутбук, который содержит следующие разделы:
  1. Текстовое описание выбранного Вами набора данных.
  2. Основные характеристики датасета.
  3. Визуальное исследование датасета.
  4. Информация о корреляции признаков.
- Сформировать отчет и разместить его в своем репозитории на github.

Дополнительно примеры решения задач, содержащие визуализацию, можно посмотреть в репозитории курса mlcourse.ai
- https://github.com/Yorko/mlcourse.ai/wiki/Individual-projects-and-tutorials-(in-Russian)

Ход работы

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

# Будем использовать только обучающую выборку
data = pd.read_csv('data/lab2_prepared_2.csv', sep=",")

# размер набора данных
data.shape
```

```
(234, 12)
```

```
# типы колонок
data.dtypes
```

```
Country or Dependency     object
Population(2020)          object
Yearly Change            float64
Net Change                object
Density(p/km^2)          float64
Land Area(km^2)           object
Migrants                 float64
Fert Rate                float64
Med Age                  float64
Urban                    float64
World Share              float64
Regions                   object
dtype: object
```

```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

```
Country or Dependency     0
Population(2020)          0
Yearly Change             0
Net Change                0
Density(p/km^2)           0
Land Area(km^2)           0
Migrants                 33
Fert Rate                33
Med Age                  33
Urban                    13
World Share               0
Regions                  11
dtype: int64
```

```
# Первые 5 строк датасета
data.head()
```

```
  Country or Dependency Population(2020)  Yearly Change  Net Change  \
0                 China    1.439.323.776           0.39   5.540.090
1                 India    1.380.004.385           0.99  13.586.631
```

```
2             Indonesia      273.523.615           1.07   2.898.047
3              Pakistan      220.892.340           2.00   4.327.022
4            Bangladesh      164.689.383           1.01   1.643.222

   Density(p/km^2) Land Area(km^2)  Migrants  Fert Rate  Med Age  Urban  \
0          153.000       9.388.211  -348.399        1.7     38.0   61.0
1          464.000       2.973.190  -532.687        2.2     28.0   35.0
2          151.000       1.811.570   -98.955        2.3     30.0   56.0
3          287.000         770.880  -233.379        3.6     23.0   35.0
4            1.265         130.170  -369.501        2.1     28.0   39.0

   World Share Regions
0        18.47     NaN
1        17.70    Asia
2         3.51    Asia
3         2.83    Asia
4         2.11    Asia
```

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

```
Всего строк: 234
```

Обработка пропусков в данных

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

```
((234, 12), (234, 7))
```

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

```
((234, 12), (190, 12))
```

```
data.head()
```

```
  Country or Dependency Population(2020)  Yearly Change  Net Change  \
0                 China    1.439.323.776           0.39   5.540.090
1                 India    1.380.004.385           0.99  13.586.631
2             Indonesia      273.523.615           1.07   2.898.047
3              Pakistan      220.892.340           2.00   4.327.022
4            Bangladesh      164.689.383           1.01   1.643.222

   Density(p/km^2) Land Area(km^2)  Migrants  Fert Rate  Med Age  Urban  \
0          153.000       9.388.211  -348.399        1.7     38.0   61.0
1          464.000       2.973.190  -532.687        2.2     28.0   35.0
2          151.000       1.811.570   -98.955        2.3     30.0   56.0
3          287.000         770.880  -233.379        3.6     23.0   35.0
4            1.265         130.170  -369.501        2.1     28.0   39.0

   World Share Regions
0        18.47     NaN
1        17.70    Asia
2         3.51    Asia
```

```
3         2.83    Asia
4         2.11    Asia
```

```
# Заполнение всех пропущенных значений нулями
# В данном случае это некорректно, так как нулями заполняются в том числе категориальные колонки
data_new_3 = data.fillna(0)
data_new_3.head()
```

```
  Country or Dependency Population(2020)  Yearly Change  Net Change  \
0                 China    1.439.323.776           0.39   5.540.090
1                 India    1.380.004.385           0.99  13.586.631
2             Indonesia      273.523.615           1.07   2.898.047
3              Pakistan      220.892.340           2.00   4.327.022
4            Bangladesh      164.689.383           1.01   1.643.222

   Density(p/km^2) Land Area(km^2)  Migrants  Fert Rate  Med Age  Urban  \
0          153.000       9.388.211  -348.399        1.7     38.0   61.0
1          464.000       2.973.190  -532.687        2.2     28.0   35.0
2          151.000       1.811.570   -98.955        2.3     30.0   56.0
3          287.000         770.880  -233.379        3.6     23.0   35.0
4            1.265         130.170  -369.501        2.1     28.0   39.0

   World Share Regions
0         18.47       0
1         17.70    Asia
2          3.51    Asia
3          2.83    Asia
4          2.11    Asia
```

Обработка пропусков в числовых данных

```
# Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.
'.format(col, dt, temp_null_count, temp_perc))
```

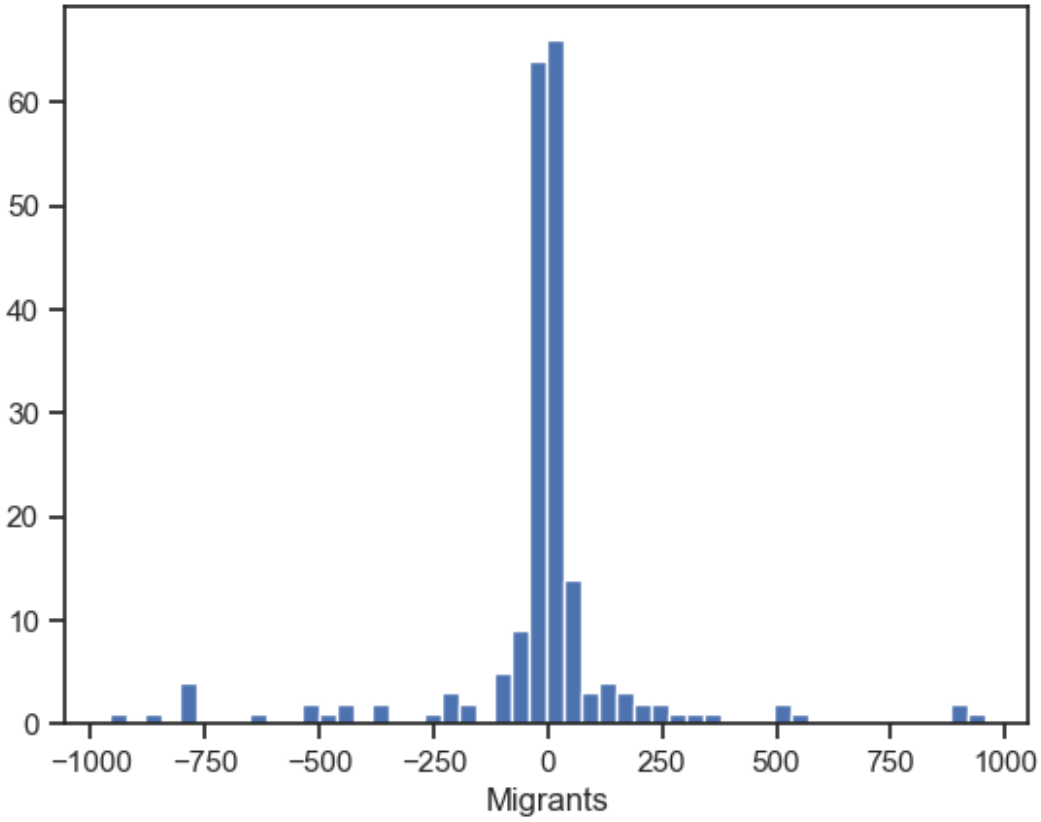
```
Колонка Migrants. Тип данных float64. Количество пустых значений 33, 14.1%.
Колонка Fert Rate. Тип данных float64. Количество пустых значений 33, 14.1%.
Колонка Med Age. Тип данных float64. Количество пустых значений 33, 14.1%.
Колонка Urban. Тип данных float64. Количество пустых значений 13, 5.56%.
```

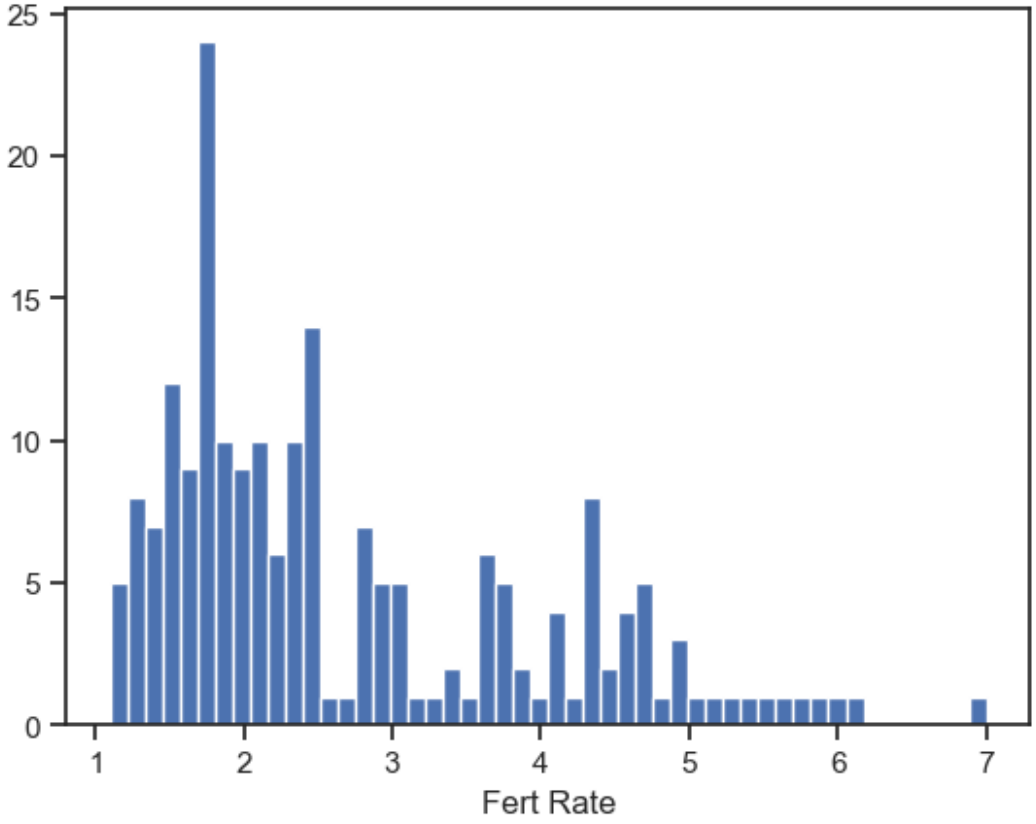
```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num
```

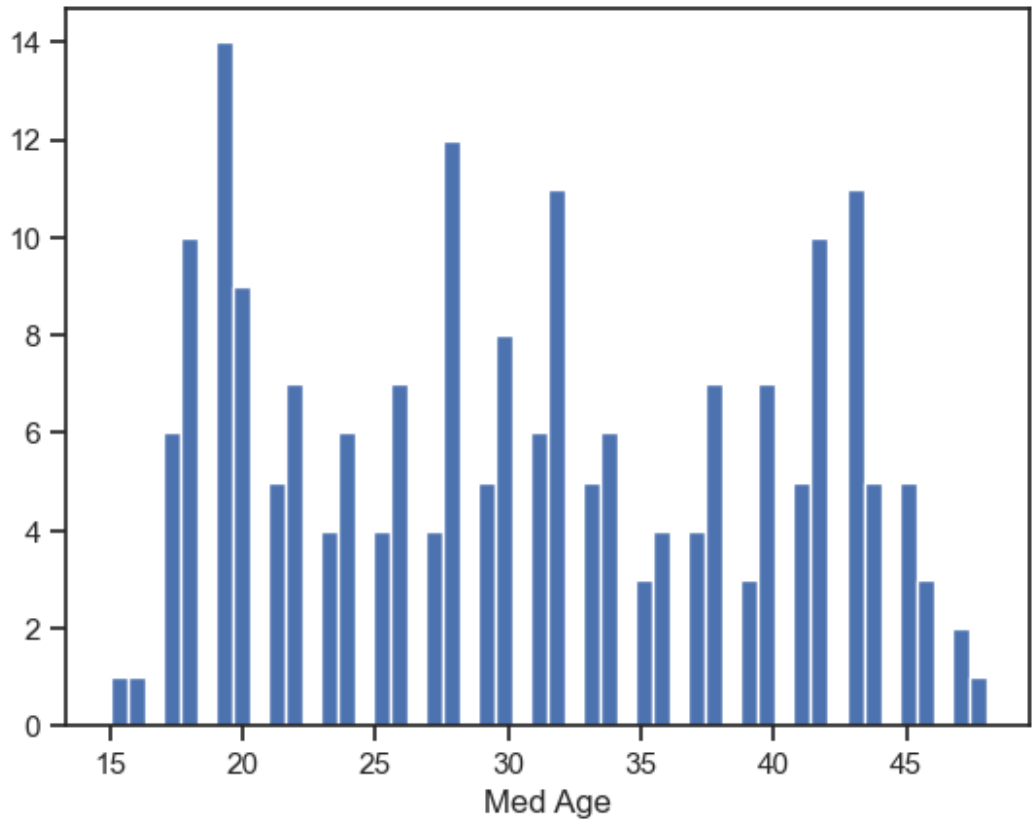
```
     Migrants  Fert Rate  Med Age  Urban
0    -348.399        1.7     38.0   61.0
1    -532.687        2.2     28.0   35.0
2     -98.955        2.3     30.0   56.0
```

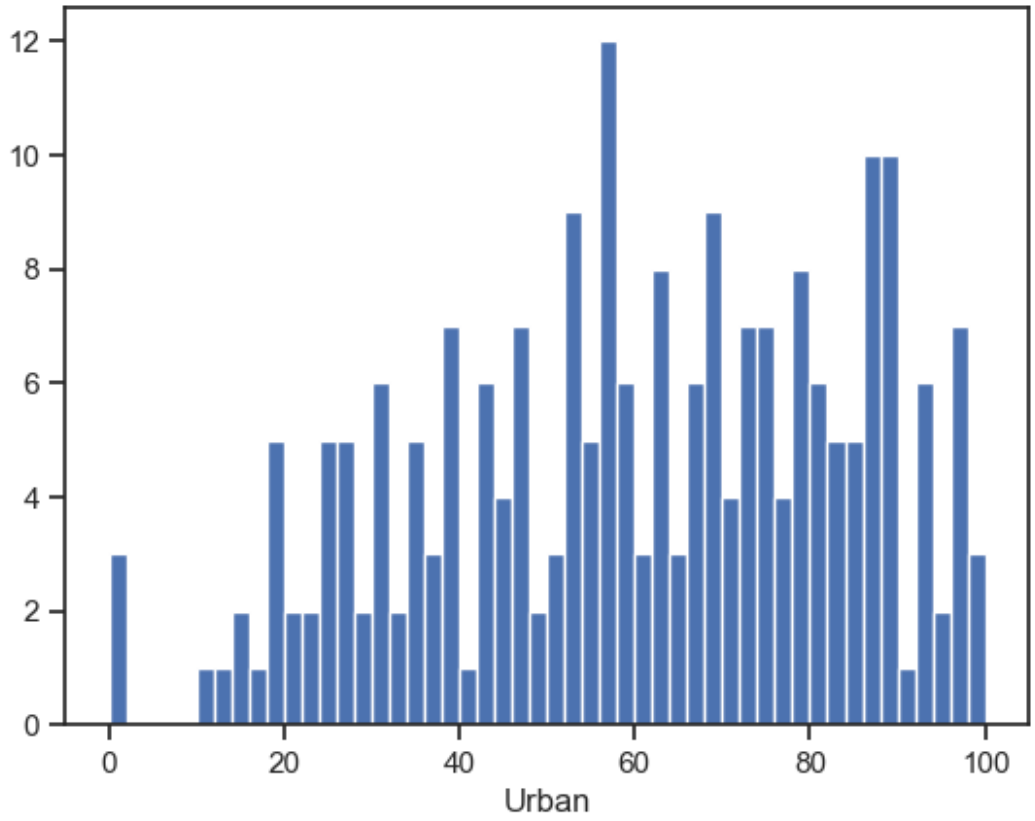
```
3    -233.379        3.6     23.0   35.0
4    -369.501        2.1     28.0   39.0
..        ...        ...      ...    ...
229       NaN        NaN      NaN   62.0
230       NaN        NaN      NaN    0.0
231       NaN        NaN      NaN    NaN
232       NaN        NaN      NaN   46.0
233       NaN        NaN      NaN    0.0

[234 rows x 4 columns]
```

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




25
20
15
10
5
0
1
2
3
4
5
6
7
Fert Rate


14
12
10
8
6
4
2
0
15
20
25
30
35
40
45
Med Age




```
data_num_Fert_Rate = data_num[['Fert Rate']]
data_num_Fert_Rate.head()
```

```
   Fert Rate
0        1.7
1        2.2
2        2.3
3        3.6
4        2.1
```

```
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_num_Fert_Rate)
mask_missing_values_only

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

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

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

```
('mean',
 array([2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398,
        2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398,
        2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398,
        2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398,
```

```
        2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398, 2.6920398,
        2.6920398, 2.6920398, 2.6920398]))
```

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

```
('median',
 array([2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3,
        2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3,
        2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3]))
```

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

```
C:\Users\prite\anaconda3\lib\site-packages\sklearn\impute\_base.py:49: Future
Warning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the defa
ult behavior of `mode` typically preserves the axis it acts along. In SciPy 1
.11.0, this behavior will change: the default value of `keepdims` will become
False, the `axis` over which the statistic is taken will be eliminated, and t
he value None will no longer be accepted. Set `keepdims` to True or False to
avoid this warning.
  mode = stats.mode(array)
```

```
('most_frequent',
 array([1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8,
        1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8,
        1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8]))
```

```
# 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_d
ata[filled_data.size-1]
```

```
data[['Urban']].describe()
```

```
            Urban
count  221.000000
mean    59.714932
std     23.818160
min      0.000000
25%     43.000000
50%     62.000000
75%     79.000000
max    100.000000
```

```
test_num_impute_col(data, 'Urban', strategies[0])
```

```
('Urban', 'mean', 13, 59.71493212669683, 59.71493212669683)
```

```
test_num_impute_col(data, 'Urban', strategies[1])
```

```
('Urban', 'median', 13, 62.0, 62.0)
```

```
test_num_impute_col(data, 'Urban', strategies[2])
```

```
C:\Users\prite\anaconda3\lib\site-packages\sklearn\impute\_base.py:49: Future
Warning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the defa
ult behavior of `mode` typically preserves the axis it acts along. In SciPy 1
.11.0, this behavior will change: the default value of `keepdims` will become
False, the `axis` over which the statistic is taken will be eliminated, and t
he value None will no longer be accepted. Set `keepdims` to True or False to
avoid this warning.
  mode = stats.mode(array)
```

```
('Urban', 'most_frequent', 13, 57.0, 57.0)
```

Обработка пропусков в категориальных данных

```
# Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета
cat_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='object'):
        cat_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.
'.format(col, dt, temp_null_count, temp_perc))
```

```
Колонка Regions. Тип данных object. Количество пустых значений 11, 4.7%.
```

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

```
  Regions
0     NaN
1    Asia
2    Asia
3    Asia
4    Asia
```

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

```
array([nan, 'Asia', 'Africa', 'Europe', 'Latin America & Caribbean',
       'Northern America', 'Oceania'], dtype=object)
```

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

```
(11, 1)
```

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

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

```
array(['Africa', 'Asia', 'Europe', 'Latin America & Caribbean',
       'Northern America', 'Oceania'], dtype=object)
```

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


np.unique(data_imp3)
```

```
array(['Africa', 'Asia', 'Europe', 'Latin America & Caribbean', 'NA',
       'Northern America', 'Oceania'], dtype=object)
```

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

```
11
```

Преобразование категориальных признаков в числовые

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

```
          c1
0     Africa
1       Asia
2       Asia
3       Asia
4       Asia
..       ...
229  Oceania
230   Africa
231  Oceania
232  Oceania
233   Africa

[234 rows x 1 columns]
```

Кодирование категорий целочисленными значениями (label encoding)

```
from sklearn.preprocessing import LabelEncoder

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

```
array(['Africa', 'Asia', 'Europe', 'Latin America & Caribbean',
       'Northern America', 'Oceania'], dtype=object)
```

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

# Наименования категорий в соответствии с порядковыми номерами

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

le.classes_
```

```
array(['Africa', 'Asia', 'Europe', 'Latin America & Caribbean',
       'Northern America', 'Oceania'], dtype=object)
```

```
cat_enc_le
```

```
array([0, 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, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 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, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2,
       2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 0, 3, 0, 4, 0, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 0, 5, 5, 0, 5, 5, 0])
```

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

```
array([0, 1, 2, 3, 4, 5])
```

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

```
array(['Africa', 'Asia', 'Europe', 'Latin America & Caribbean'],
      dtype=object)
```

Кодирование категорий наборами бинарных значений

```
from sklearn.preprocessing import OneHotEncoder

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

```
cat_enc.shape
```

```
(234, 1)
```

```
cat_enc_ohe.shape
```

```
(234, 6)
```

```
cat_enc_ohe
```

```
<234x6 sparse matrix of type '<class 'numpy.float64'>'
      with 234 stored elements in Compressed Sparse Row format>
```

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

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

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

```
       c1
0  Africa
```

```
1    Asia
2    Asia
3    Asia
4    Asia
5    Asia
6    Asia
7    Asia
8    Asia
9    Asia
```

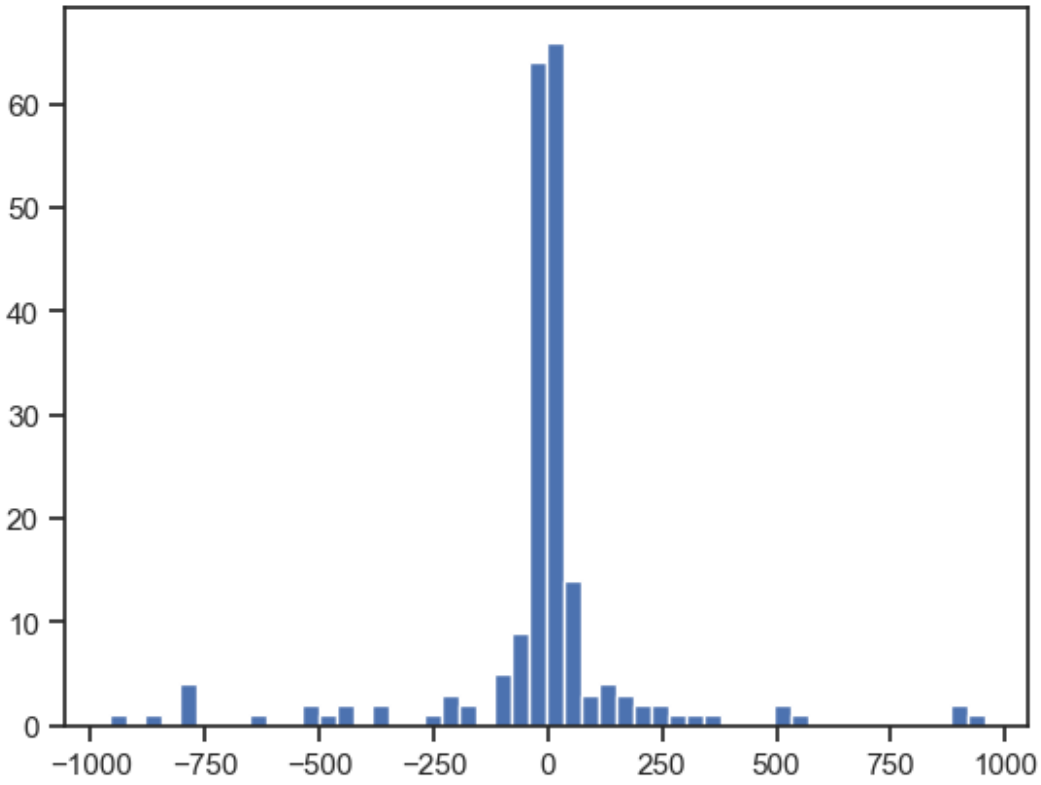
```
pd.get_dummies(cat_enc).head()
```

```
   c1_Africa  c1_Asia  c1_Europe  c1_Latin America & Caribbean  \
0          1        0          0                             0
1          0        1          0                             0
2          0        1          0                             0
3          0        1          0                             0
4          0        1          0                             0

   c1_Northern America  c1_Oceania
0                    0           0
1                    0           0
2                    0           0
3                    0           0
4                    0           0
```

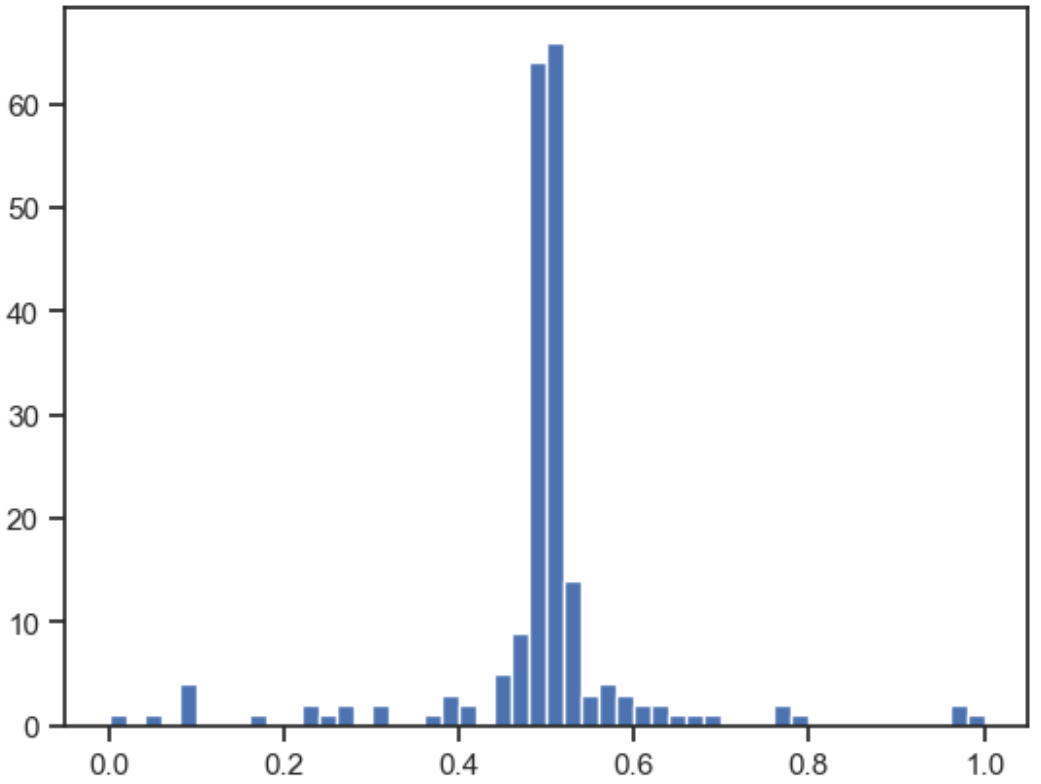
```
pd.get_dummies(cat_temp_data, dummy_na=True).head()
```

```
   Regions_Africa  Regions_Asia  Regions_Europe  \
0               0             0               0
1               0             1               0
2               0             1               0
3               0             1               0
4               0             1               0

   Regions_Latin America & Caribbean  Regions_Northern America  \
0                                  0                         0
1                                  0                         0
2                                  0                         0
3                                  0                         0
4                                  0                         0

   Regions_Oceania  Regions_nan
0                0            1
1                0            0
2                0            0
3                0            0
4                0            0
```

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

sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['Migrants']])

plt.hist(data['Migrants'], 50)
plt.show()
```




```
plt.hist(sc1_data, 50)
plt.show()
```



Масштабирование данных на основе Z-оценки

```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Migrants']])
```

```
plt.hist(sc2_data, 50)
plt.show()
```