

Защищено:  
Гапанюк Ю.Е.

Демонстрация:  
Гапанюк Ю.Е.

"\_\_" \_\_\_\_\_ 2022 г.

"\_\_" \_\_\_\_\_ 2022 г.

**Отчет по лабораторной работе № 2 по курсу  
Технологии машинного обучения  
ГУИМЦ**

**Тема работы: " Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных. "**

13  
(количество листов)  
Вариант № 4

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-84Б

Шанаурина Е. Г.

\_\_\_\_\_  
(подпись)

"\_\_" \_\_\_\_\_ 2022 г.

# Цель лабораторной работы

Изучение способов предварительной обработки данных для дальнейшего формирования моделей.

## 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.

## Ход выполнения работы

### Текстовое описание набора данных

В качестве набора данных используется dataset рейтингов университетов мира на основании трёх рейтингов. Датасет доступен по адресу: <https://www.kaggle.com/mylesoneill/world-university-rankings>

Из набора данных будет рассматриваться только файл `cwurData.csv`.

Описание столбцов:

- `world_rank` - мировой рейтинг университета
- `institution` - название университета
- `country` - страна, в которой расположен университет
- `national_rank` - рейтинг университета в стране его нахождения
- `quality_of_education` - рейтинг качества образования
- `quality_of_faculty` - рейтинг качества профессорско-преподавательского состава
- `publications` - рейтинг публикаций
- `infuence` - рейтинг влияния
- `citations` - количество студентов в университете
- `broad_impact` - рейтинг за широкое влияние (предоставлен только за 2014 и 2015 гг. Остальное - пропуски)
- `patents` - рейтинг за патенты
- `score` - общий балл, используемый для определения мирового рейтинга
- `year` - год рейтинга (с 2012 по 2015 год)

### Основные характеристики набора данных

Подключаем все необходимые библиотеки

```
In [1]:
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
import matplotlib_inline
```

```
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

## Подключаем Dataset

In [2]:

```
data = pd.read_csv('cwurData.csv', sep=",")
```

## Размер набора данных

In [3]:

```
data.shape
```

Out[3]:

```
(2200, 14)
```

## Типы колонок

In [4]:

```
data.dtypes
```

Out[4]:

```
world_rank          int64
institution          object
country             object
national_rank       int64
quality_of_education int64
alumni_employment   int64
quality_of_faculty   int64
publications        int64
influence            int64
citations            int64
broad_impact        float64
patents             int64
score               float64
year                int64
dtype: object
```

## Проверяем, есть ли пропущенные значения

In [5]:

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

Out[5]:

```
world_rank          0
institution          0
country             0
national_rank       0
quality_of_education 0
alumni_employment   0
quality_of_faculty   0
publications        0
influence            0
citations            0
broad_impact        200
patents             0
score               0
year                0
dtype: int64
```

## Первые 5 строк датасета

In [6]:

```
data.head()
```

Out[6]:

	world_rank	institution	country	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publication
0	1	Harvard University	USA	1	7	9	1	
1	2	Massachusetts Institute of Technology	USA	2	9	17	3	1
2	3	Stanford University	USA	3	17	11	5	
3	4	University of Cambridge	United Kingdom	1	10	24	4	1
4	5	California Institute of Technology	USA	4	2	29	7	3

In [7]:

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

Всего строк: 2200

Процент пропусков в `broad_impact`

In [8]:

```
(200 / 2200) * 100
```

Out[8]:

9.090909090909092

Настройка отображения графиков

In [9]:

```
# Задание формата графиков для сохранения высокого качества PNG
from IPython.display import set_matplotlib_formats
matplotlib_inline.backend_inline.set_matplotlib_formats("retina")
# Задание ширины графиков, чтобы они помещались на A4
```

## Обработка пропусков данных

### Очистка строк

Можно очистить строки, содержащие пропуски. При этом останутся данные только за 2014 и 2015 гг (см. описание датасета)

In [10]:

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

Out[10]:

((2200, 14), (2000, 14))

Выведем первые 11 строк, чтобы убедиться, что данные в `national_rank` числовые (Jupyter Lab в предпросмотре CSV показывает не совсем верно)

In [11]:

```
data_no_null.head(11)
```

Out[11]:

	world_rank	institution	country	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publicatic
200	1	Harvard University	USA	1	1	1	1	
201	2	Stanford University	USA	2	11	2	4	
202	3	Massachusetts Institute of Technology	USA	3	3	11	2	
203	4	University of Cambridge	United Kingdom	1	2	10	5	
204	5	University of Oxford	United Kingdom	2	7	12	10	
205	6	Columbia University	USA	4	13	8	9	
206	7	University of California, Berkeley	USA	5	4	22	6	
207	8	University of Chicago	USA	6	10	14	8	
208	9	Princeton University	USA	7	5	16	3	
209	10	Yale University	USA	8	9	25	11	
210	11	Cornell University	USA	9	12	18	19	

In [12]:

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

Всего строк: 2000

## Внедрение значений

In [13]:

```
# 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 `broad_impact`. Тип данных `float64`. Количество пустых значений 200, 10.0%.

In [33]:

```
Фильтр по колонкам с пропушенными значениями
```

```
data_num = data[num_cols]
data_num
```

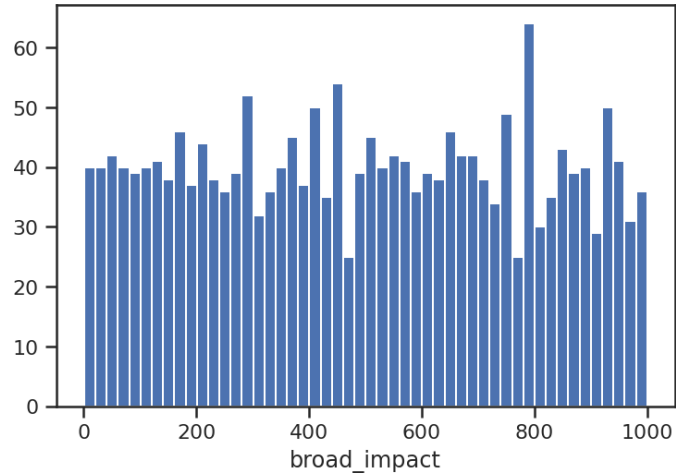
Out[33]:

broad_impact	
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
...	...
2195	969.0
2196	981.0
2197	975.0
2198	975.0
2199	981.0

2200 rows x 1 columns

In [34]:

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



Будем использовать встроенные средства импьютации библиотеки scikit-learn - <https://scikit-learn.org/stable/modules/impute.html>

In [35]:

```
data_num_MasVnrArea = data_num[['broad_impact']]
data_num_MasVnrArea.head()
```

Out[35]:

broad_impact	
0	NaN
1	NaN
2	NaN
3	NaN

In [36]:

In [37]:

Out[37]:

С помощью класса `SimpleImputer` проверим импутацию различными [показателями центра распределения](#)

In [38]:

In [39]:

In [40]:

Out[40]:

[illegible]

In [41]:



```
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 [44]:

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

Out[44]:

broad_impact	
count	2000.000000
mean	496.699500
std	286.919755
min	1.000000
25%	250.500000
50%	496.000000
75%	741.000000
max	1000.000000

In [47]:

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

Out[47]:

```
('broad_impact', 'mean', 200, 496.6995, 496.6995)
```

In [48]:

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

Out[48]:

```
('broad_impact', 'median', 200, 496.0, 496.0)
```

In [50]:

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

Out[50]:

```
('broad_impact', 'most_frequent', 200, 642.0, 642.0)
```

## Кодирование категориальных признаков

Преобразуем названия стран, городов, ... в числовые значения (label encoding)

In [14]:

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

```
=====> institution <=====
```

In [15]:

```
le = LabelEncoder()
institution_le = le.fit_transform(data_no_null['institution'])
```

In [16]:

```
data_no_null['institution'].unique()
```

Out[16]:

```
array(['Harvard University', 'Stanford University',  
      'Massachusetts Institute of Technology', ...,  
      'Babeş-Bolyai University', 'Henan Normal University',  
      'Southwest Jiaotong University'], dtype=object)
```

In [17]:

```
arr_institution_encoded = np.unique(institution_le)  
arr_institution_encoded
```

Out[17]:

```
array([ 0, 1, 2, ..., 1020, 1021, 1022])
```

In [18]:

```
le.inverse_transform([n for n in range(1023)])
```

Out[18]:

```
array(['AGH University of Science and Technology', 'Aalborg University',  
      'Aalto University', ..., 'École normale supérieure de Cachan',  
      'École normale supérieure de Lyon', 'Örebro University'],  
      dtype=object)
```

=====> country <=====

In [19]:

```
le_country = LabelEncoder()  
country_le = le_country.fit_transform(data_no_null['country'])
```

In [20]:

```
data_no_null['country'].unique()
```

Out[20]:

```
array(['USA', 'United Kingdom', 'Japan', 'Switzerland', 'Israel',  
      'South Korea', 'Canada', 'France', 'Russia', 'China', 'Taiwan',  
      'Sweden', 'Singapore', 'Denmark', 'Germany', 'Netherlands',  
      'Italy', 'Belgium', 'Australia', 'Finland', 'Norway',  
      'South Africa', 'Spain', 'Brazil', 'Hong Kong', 'Ireland',  
      'Austria', 'New Zealand', 'Portugal', 'Thailand', 'Czech Republic',  
      'Malaysia', 'India', 'Greece', 'Mexico', 'Hungary', 'Argentina',  
      'Turkey', 'Poland', 'Saudi Arabia', 'Chile', 'Iceland', 'Slovenia',  
      'Estonia', 'Lebanon', 'Croatia', 'Colombia', 'Slovak Republic',  
      'Iran', 'Egypt', 'Serbia', 'Bulgaria', 'Lithuania', 'Uganda',  
      'United Arab Emirates', 'Uruguay', 'Cyprus', 'Romania',  
      'Puerto Rico'], dtype=object)
```

In [21]:

```
np.unique(country_le)
```

Out[21]:

```
array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,  
      17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,  
      34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,  
      51, 52, 53, 54, 55, 56, 57, 58])
```

In [22]:

```
le_country.inverse_transform([n for n in range(59)])
```

Out[22]:

```
array(['Argentina', 'Australia', 'Austria', 'Belgium', 'Brazil',  
      'Bulgaria', 'Canada', 'Chile', 'China', 'Colombia', 'Croatia',  
      'Cyprus', 'Czech Republic', 'Denmark', 'Egypt', 'Estonia',  
      'Finland', 'France', 'Germany', 'Greece', 'Hong Kong', 'Hungary',
```

```
'Iceland', 'India', 'Iran', 'Ireland', 'Israel', 'Italy', 'Japan',
'Lebanon', 'Lithuania', 'Malaysia', 'Mexico', 'Netherlands',
'New Zealand', 'Norway', 'Poland', 'Portugal', 'Puerto Rico',
'Romania', 'Russia', 'Saudi Arabia', 'Serbia', 'Singapore',
'Slovak Republic', 'Slovenia', 'South Africa', 'South Korea',
'Spain', 'Sweden', 'Switzerland', 'Taiwan', 'Thailand', 'Turkey',
'USA', 'Uganda', 'United Arab Emirates', 'United Kingdom',
'Uruguay']], dtype=object)
```

In [23]:

```
data_no_null.head()
```

Out[23]:

	world_rank	institution	country	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications
200	1	Harvard University	USA	1	1	1	1	
201	2	Stanford University	USA	2	11	2	4	
202	3	Massachusetts Institute of Technology	USA	3	3	11	2	
203	4	University of Cambridge	United Kingdom	1	2	10	5	
204	5	University of Oxford	United Kingdom	2	7	12	10	

In [24]:

```
data_digit = data_no_null.copy()
#data_digit.pop('institution')
#data_digit.pop('country')
data_digit["institution"] = institution_le
data_digit['country'] = country_le
data_digit
```

Out[24]:

	world_rank	institution	country	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications	i
200	1	184	54	1	1	1	1	1	
201	2	511	54	2	11	2	4	5	
202	3	312	54	3	3	11	2	15	
203	4	637	57	1	2	10	5	10	
204	5	819	57	2	7	12	10	11	
...	...	...	...	...	...	...	...	...	...
2195	996	954	37	7	367	567	218	926	
2196	997	11	14	4	236	566	218	997	
2197	998	132	4	18	367	549	218	830	
2198	999	576	48	40	367	567	218	886	
2199	1000	74	8	83	367	567	218	861	

2000 rows x 14 columns

Проверяем типы данных

In [25]:

```
data_digit.dtypes
```

```
Out[25]:
```

```
world_rank          int64
institution          int64
country             int64
national_rank       int64
quality_of_education int64
alumni_employment   int64
quality_of_faculty   int64
publications        int64
influence           int64
citations           int64
broad_impact        float64
patents             int64
score               float64
year               int64
dtype: object
```

## Масштабирование данных

Масштабирование будем проводить на `data_digit` (где нет категориальных признаков)

```
In [26]:
```

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

### MinMax масштабирование

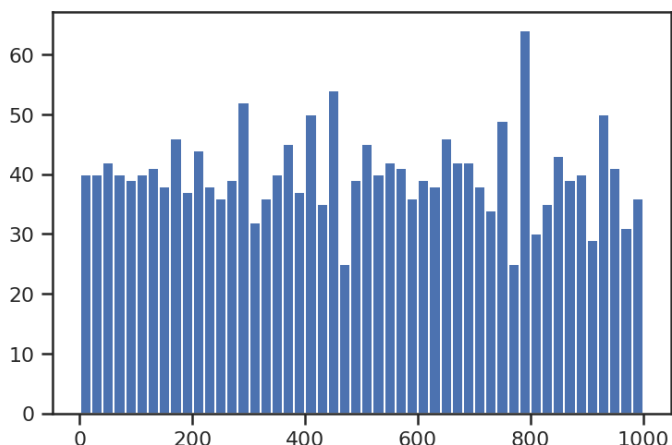
```
=====> world_rank <=====
```

```
In [27]:
```

```
scl = MinMaxScaler()
scl_data = scl.fit_transform(data_digit[['broad_impact']])
```

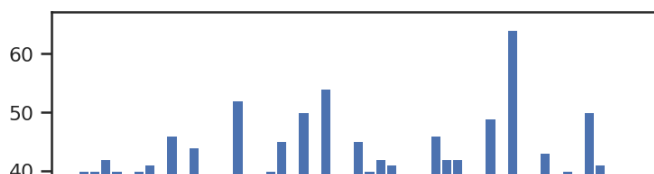
```
In [28]:
```

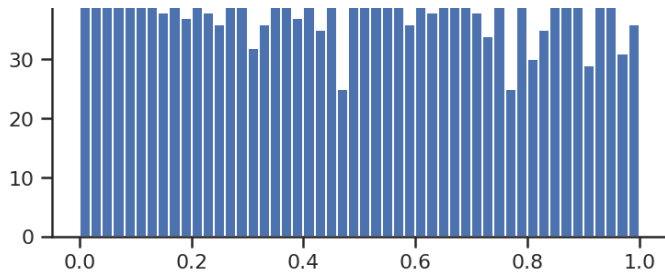
```
plt.hist(data_digit['broad_impact'], 50)
plt.show()
```



```
In [29]:
```

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





In [ ]:

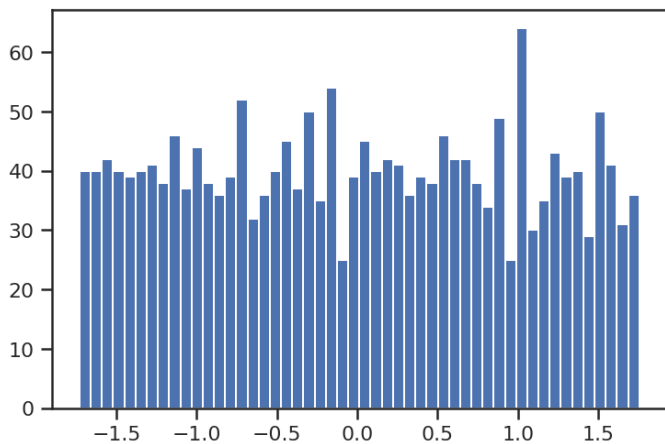
## Масштабирование данных на основе Z-оценки - StandardScaler

In [30]:

```
sc2 = StandardScaler()  
sc2_data = sc2.fit_transform(data_digit[['broad_impact']])
```

In [31]:

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



In [ ]: