

## Рубежный контроль №1

### Тема: Технологии разведочного анализа и обработки данных.

Вариант 18. Номер задачи №3. Номер набора данных №2.

Задача №3. Для заданного набора данных произведите масштабирование данных (для одного признака) и преобразование категориальных признаков в количественные двумя способами (label encoding, one hot encoding) для одного признака.

#### Текстовое описание набора данных

В качестве набора данных мы будем использовать набор данных [FiveThirtyEight Comic Characters Dataset](#)

Набор данных содержит колонки:

- page\_id - уникальный идентификатор страницы с персонажами в вики
- name - имя персонажа
- urlslug - уникальный URL-адрес вики, который приведет вас к персонажу
- ID - статус личности персонажа
- ALIGN - персонаж хороший, плохой или нейтральный
- EYE - цвет глаз персонажа
- HAIR - цвет волос персонажа
- SEX - пол персонажа (например, мужской, женский и т. д.)
- GSM - персонаж принадлежит к полу или сексуальному меньшинству (например, гомосексуальные персонажи, бисексуальные персонажи)
- ALIVE - персонаж жив или умер
- APPEARANCES - количество появлений персонажа в комиксах (по состоянию на 2 сентября 2014 г.)
- FIRST APPEARANCE - месяц и год первого появления персонажа в комиксе, если таковой имеется
- YEAR - год первого появления персонажа в комиксе, если таковой имеется

#### Импорт библиотек и загрузка данных

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

```
In [2]: # Будем анализировать данные файла Future50
data = pd.read_csv('data/dc-wikia-data.csv', sep=",")
```

## Основные характеристики датасета

```
In [3]: # Первые 5 строк датасета
data.head()
```

Out[3]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX
	1422	Batman (Bruce Wayne)	VwikiVBatman_(Bruce_Wayne)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters
	23387	Superman (Clark Kent)	VwikiVSuperman_(Clark_Kent)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters
	1458	Green Lantern (Hal Jordan)	VwikiVGreen_Lantern_(Hal_Jordan)	Secret Identity	Good Characters	Brown Eyes	Brown Hair	Male Characters
	1659	James Gordon (New Earth)	VwikiVJames_Gordon_(New_Earth)	Public Identity	Good Characters	Brown Eyes	White Hair	Male Characters
	1576	Richard Grayson (New Earth)	VwikiVRichard_Grayson_(New_Earth)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters

```
In [4]: # Размер датасета
data.shape
```

Out[4]: (6896, 13)

```
In [5]: # Список колонок с типами данных
data.dtypes
```

```
Out[5]: page_id      int64
name              object
urlslug          object
ID               object
ALIGN            object
EYE              object
HAIR             object
SEX              object
GSM              object
ALIVE            object
APPEARANCES      float64
FIRST APPEARANCE object
YEAR             float64
dtype: object
```

```
In [8]: # Проверим наличие пустых значений
data.isnull().sum()
```

```
Out[8]: page_id      0
name              0
urlslug           0
ID               2013
ALIGN            601
EYE              3628
HAIR             2274
SEX              125
GSM              6832
ALIVE             3
APPEARANCES      355
FIRST APPEARANCE  69
YEAR             69
dtype: int64
```

```
B [9]: # Основные статистические характеристики набора данных
data.describe()
```

Out[9]:

	page_id	APPEARANCES	YEAR
count	6896.000000	6541.000000	6827.000000
mean	147441.209252	23.625134	1989.766662
std	108388.631149	87.378509	16.824194
min	1380.000000	1.000000	1935.000000
25%	44105.500000	2.000000	1983.000000
50%	141267.000000	6.000000	1992.000000
75%	213203.000000	15.000000	2003.000000
max	404010.000000	3093.000000	2013.000000

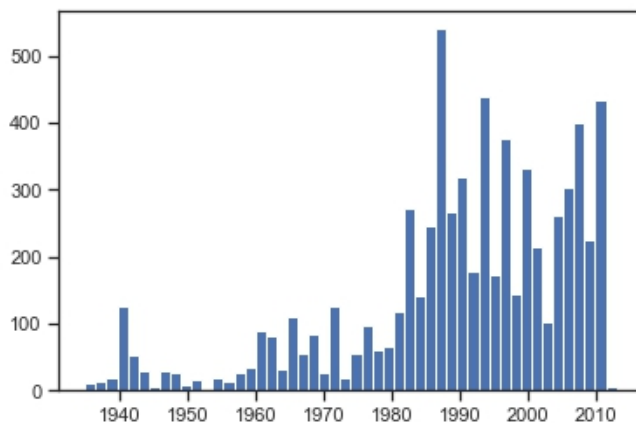
### Масштабирование данных (для одного признака)

```
B [10]: # импорт
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

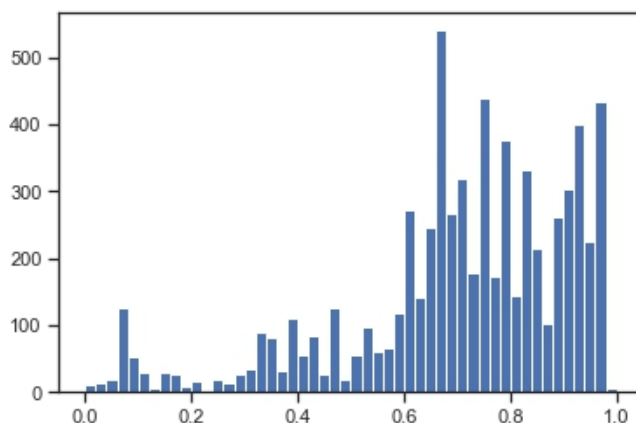
#### - MinMax масштабирование

```
B [11]: sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['YEAR']])
```

```
B [14]: plt.hist(data['YEAR'], 50)
plt.show()
```



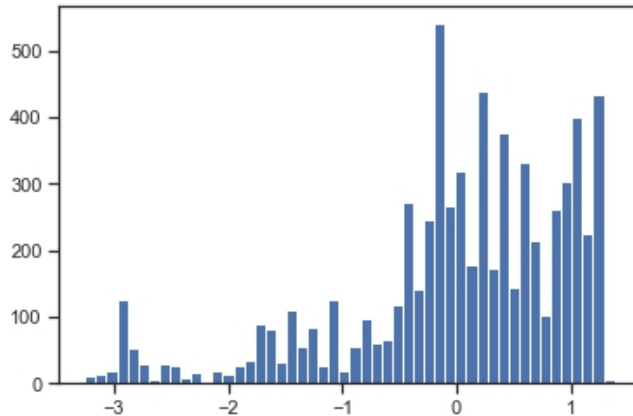
```
B [15]: plt.hist(sc1_data, 50)
plt.show()
```



### - Масштабирование данных на основе Z-оценки

```
B [16]: sc2 = StandardScaler()  
sc2_data = sc2.fit_transform(data[['YEAR']])
```

```
B [17]: plt.hist(sc2_data, 50)  
plt.show()
```



### Преобразование категориальных признаков в количественные (label encoding, one hot encoding) для одного признака

```
B [18]: cat_temp_data = data[['SEX']]
```

```
B [19]: cat_temp_data['SEX'].unique()
```

```
Out[19]: array(['Male Characters', 'Female Characters', nan,  
               'Genderless Characters', 'Transgender Characters'], dtype=object)
```

```
B [20]: #обработка пропусков  
from sklearn.impute import SimpleImputer
```

```
B [21]: # Импутация наиболее частыми значениями  
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')  
data_imp2 = imp2.fit_transform(cat_temp_data)  
np.unique(data_imp2)
```

```
Out[21]: array(['Female Characters', 'Genderless Characters', 'Male Characters',  
               'Transgender Characters'], dtype=object)
```

### - label encoding

```
B [22]: cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
cat_enc
```

Out[22]:

	c1
0	Male Characters
1	Male Characters
2	Male Characters
3	Male Characters
4	Male Characters
...	...
6891	Female Characters
6892	Male Characters
6893	Male Characters
6894	Male Characters
6895	Male Characters

6896 rows x 1 columns

```
B [23]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
B [28]: le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
```

```
B [25]: np.unique(cat_enc_le)
```

Out[25]: array([0, 1, 2, 3])

```
B [32]: le.inverse_transform([ 0, 1, 2, 3])
```

Out[32]: array(['Female Characters', 'Genderless Characters', 'Male Characters',  
'Transgender Characters'], dtype=object)

### - one-hot encoding

```
B [33]: ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])
```

```
B [34]: cat_enc.shape
```

Out[34]: (6896, 1)

```
B [35]: cat_enc_ohe.shape
```

Out[35]: (6896, 4)

```
B [36]: cat_enc_ohe.todense()[0:10]
```

Out[36]: matrix([[0., 0., 1., 0.],  
[0., 0., 1., 0.],  
[0., 0., 1., 0.],  
[0., 0., 1., 0.],  
[0., 0., 1., 0.],  
[1., 0., 0., 0.],  
[0., 0., 1., 0.],  
[0., 0., 1., 0.],  
[1., 0., 0., 0.],  
[0., 0., 1., 0.]])



```
B [37]: cat_enc.head(10)
```

Out[37]:

	c1
0	Male Characters
1	Male Characters
2	Male Characters
3	Male Characters
4	Male Characters
5	Female Characters
6	Male Characters
7	Male Characters
8	Female Characters
9	Male Characters

### Гистограмма

Позволяет оценить плотность вероятности распределения данных.

```
B [41]: fig, ax = plt.subplots(figsize=(5,5))
sns.distplot(data['YEAR'])
```

C:\Users\kenia\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

Out[41]: <AxesSubplot:xlabel='YEAR', ylabel='Density'>

