**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)

О Т Ч Е Т

**по лабораторной работе**

по дисциплине: Технологии машинного обучения

на тему: Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных

\_

\_

\_

\_

\_

\_

\_

\_

Студент РТ5-61Б **А.C. Акушко**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель

**Ю.Е. Гапанюк**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

# Лабораторная работа №3

**Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных**

## Цель лабораторной работы

### 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.

In [140]:

**import numpy as np import pandas as pd import seaborn as sns**

**import matplotlib.pyplot as plt**

%**matplotlib** inline sns.set(style="ticks")

# Загрузка данных

Ссылка на датасет: [https://www.kaggle.com/fivethirtyeight/fivethirtyeight-comic-characters-](https://www.kaggle.com/fivethirtyeight/fivethirtyeight-comic-characters-dataset) [dataset](https://www.kaggle.com/fivethirtyeight/fivethirtyeight-comic-characters-dataset)

In [192]:

data = pd.read\_csv('data/marvel-wikia-data.csv', sep=",") data.head()

Out[192]:

**page\_id name urlslug ID ALIGN EYE HAIR**

**0**

1678

Spider-

Man (Peter Parker)

\/Spider-Man\_(Peter\_Parker)

Secret

Good Hazel Brown

Identity Characters

Eyes

Hair

Captain

America Public Good Blue White

**1** 7139 (Steven Rogers)

\/Captain\_America\_(Steven\_Rogers) Identity Characters Eyes Hair

**2** 64786

Wolverine (James

\"Logan\"

Howlett)

\/Wolverine\_(James\_%22Logan%22\_Howlett) Public Neutral

Identity Characters

Blue

Eyes

Black

Hair

Iron Man

Odinson) Identity

|  |  |  |
| --- | --- | --- |
| **3** 1868 (Anthony \/Iron\_Man\_(Anthony\_%22Tony%22\_Stark) Public Good  \"Tony\" Identity Characters  Stark) | Blue Eyes | Black Hair |
| Thor No Good | Blue | Blond |
| **4** 2460 (Thor \/Thor\_(Thor\_Odinson) Dual Characters | Eyes | Hair |

In [142]:

*# размер набора данных*

data.shape

Out[142]: (16376, 13)

In [143]:

*# типы колонок*

data.dtypes

Out[143]: 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 [144]:

*# проверим есть ли пропущенные значения*

data.isnull().sum()

Out[144]: page\_id 0

name 0

urlslug 0

ID 3770

ALIGN 2812

EYE 9767

HAIR 4264

SEX 854

GSM 16286

ALIVE 3

APPEARANCES 1096

FIRST APPEARANCE 815

Year 815

dtype: int64

In [146]:

total\_count = data.shape[0]

print('Всего строк: **{}**'.format(total\_count))

Всего строк: 16376

# Обработка пропусков в данных

## Простые стратегии: удаление

In [147]:

data = data.dropna(axis=1, thresh=int(data.shape[0] \* 0.49))

In [148]:

data.isnull().sum()

Out[148]: page\_id 0

name 0

urlslug 0

ID 3770

ALIGN 2812

HAIR 4264

SEX 854

ALIVE 3

APPEARANCES 1096

FIRST APPEARANCE 815

Year 815

dtype: int64

In [149]:

data[data['ALIVE'].isnull()]

Out[149]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **page\_id** | **name** | **urlslug** | **ID** | **ALIGN** | **HAIR** | **SEX** | **ALIVE** | **APPEARANCES** |
| **16293** | 541449 | Mj7711 | \/User:Mj7711 | NaN | NaN | NaN | NaN | NaN | NaN |
| **16329** | 714409 | Sharjeel786 | \/User:Sharjeel786 | NaN | NaN | NaN | NaN | NaN | NaN |
| **16347** | 462671 | TOR\/test | \/User:TOR\/test | NaN | NaN | NaN | NaN | NaN | NaN |

In [150]:

*# Удаление 3 строк*

data = data.drop(data.index[[16293,16329,16347]])

In [151]:

data.isnull().sum()

Out[151]: page\_id 0

name 0

urlslug 0

ID 3767

ALIGN 2809

HAIR 4261

SEX 851

ALIVE 0

APPEARANCES 1093

FIRST APPEARANCE 812

Year 812

dtype: int64

## "Внедрение значений" - импьютация (imputation)

### Обработка пропусков в числовых данных

In [152]:

*# 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\_n ull\_count, temp\_perc))

Колонка APPEARANCES. Тип данных float64. Количество пустых значений 1093, 6.67%.

Колонка Year. Тип данных float64. Количество пустых значений 812, 4.96%.

In [153]:

*# Фильтр по колонкам с пропущенными значениями*

data\_num = data[num\_cols] data\_num

|  |  |  |  |
| --- | --- | --- | --- |
| Out[153]: |  | | |
|  |  | **APPEARANCES** | **Year** |
|  | **0** | 4043.0 | 1962.0 |
|  | **1** | 3360.0 | 1941.0 |
|  | **2** | 3061.0 | 1974.0 |
|  | **3** | 2961.0 | 1963.0 |
|  | **4** | 2258.0 | 1950.0 |
|  | **...** | ... | ... |
|  | **16371** | NaN | NaN |
|  | **16372** | NaN | NaN |
|  | **16373** | NaN | NaN |
|  | **16374** | NaN | NaN |
|  | **16375** | NaN | NaN |

16373 rows × 2 columns

In [154]:

*# Гистограмма по признакам*

**for** col **in** data\_num: plt.hist(data[col], 50) plt.xlabel(col) plt.show()

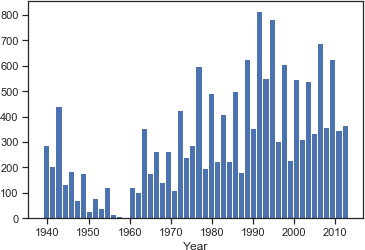
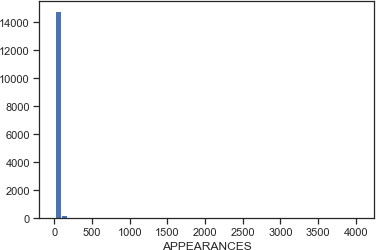
c:\users\user\appdata\local\programs\python\python37-32\lib\site-package s\numpy\lib\histograms.py:839: RuntimeWarning: invalid value encountered

in greater\_equal

keep = (tmp\_a >= first\_edge) c:\users\user\appdata\local\programs\python\python37-32\lib\site-package

s\numpy\lib\histograms.py:840: RuntimeWarning: invalid value encountered in less\_equal

keep &= (tmp\_a <= last\_edge)



In [155]:

*# Фильтр по пустым значениям поля APPEARANCES*

data[data['APPEARANCES'].isnull()]

Out[155]:

**page\_id name urlslug ID ALIGN HAIR SE**

**15280**

Minister of 743309 Castile D'or

(Earth-616)

\/Minister\_of\_Castile\_D%27or\_(Earth-

616)

No Dual

Identity

Neutral Mal

Characters NaN Character

**15281** 645438

Mr. Harris' Secretary (Earth-616)

\/Mr.\_Harris%27\_Secretary\_(Earth-

616)

No Dual Identity

Neutral Characters

Blond Hair

Femal Character

**15282** 331151 N'Jaga

(Earth-616)

No

\/N%27Jaga\_(Earth-616) Dual

Identity

Bad

Characters

NaN Mal

Character

**15283** 505986 Ertve

(Earth-616)

\/Ertve\_(Earth-616) Secret

Identity

Good Characters

White Hair

Mal Character

**15284** 19657

Invisible

Man (Gade) (Earth-616)

\/Invisible\_Man\_(Gade)\_(Earth-616)

Secret

Good

Identity Characters

NaN Mal

Character

**...** ... ... ... ... ... ... .

**16371** 657508 Ru'ach

(Earth-616)

No

\/Ru%27ach\_(Earth-616) Dual

Identity

Bad

Characters

No

Mal

Hair Character

**16372** 665474

Thane (Thanos'

\/Thane\_(Thanos%27\_son)\_(Earth-

No Dual

Good

Bald Mal

**16375** 673702

Yologarch

(Earth-616)

\/Yologarch\_(Earth-616)

NaN

Bad

Characters

NaN

Na

1093 rows × 11 columns

|  |  |  |  |
| --- | --- | --- | --- |
|  | | son) (Earth-616) | 616) Identity Characters Character |
| **16373** | 695217 | Tinkerer (Skrull) | \/Tinkerer\_(Skrull)\_(Earth-616) Secret Bad Bald Mal  Identity Characters Character |
|  |  | (Earth-616) |  |
| **16374** | 708811 | TK421  (Spiderling) | \/TK421\_(Spiderling)\_(Earth-616) Secret Neutral NaN Mal  Identity Characters Character |
|  |  | (Earth-616) |  |

In [156]:

*# Запоминаем индексы строк с пустыми значениями* flt\_index = data[data['APPEARANCES'].isnull()].index flt\_index

Out[156]: Int64Index([15280, 15281, 15282, 15283, 15284, 15285, 15286, 15287, 15288,

15289,

...

16366, 16367, 16368, 16369, 16370, 16371, 16372, 16373, 16374,

16375],

dtype='int64', length=1093)

In [157]:

*# Проверяем что выводятся нужные строки*

data[data.index.isin(flt\_index)]

Out[157]:

**page\_id name urlslug ID ALIGN HAIR SE**

**15280**

Minister of 743309 Castile D'or

(Earth-616)

\/Minister\_of\_Castile\_D%27or\_(Earth-

616)

No Dual

Identity

Neutral Mal

Characters NaN Character

**15281** 645438

Mr. Harris' Secretary (Earth-616)

\/Mr.\_Harris%27\_Secretary\_(Earth-

616)

No Dual Identity

Neutral Characters

Blond Hair

Femal Character

**15282** 331151 N'Jaga

(Earth-616)

No

\/N%27Jaga\_(Earth-616) Dual

Identity

Bad

Characters

NaN Mal

Character

**15283** 505986 Ertve

(Earth-616)

\/Ertve\_(Earth-616) Secret

Identity

Good Characters

White Hair

Mal Character

**15284** 19657

Invisible

Man (Gade) (Earth-616)

\/Invisible\_Man\_(Gade)\_(Earth-616)

Secret

Good

Identity Characters

NaN Mal

Character

**...** ... ... ... ... ... ... .

No

**16371** 657508 Ru'ach

(Earth-616)

\/Ru%27ach\_(Earth-616) Dual

Identity

Bad

Characters

No

Mal

Hair Character

**16372** 665474

Thane (Thanos'

\/Thane\_(Thanos%27\_son)\_(Earth-

No Dual

Good

Bald Mal

**16375** 673702

Yologarch

(Earth-616)

\/Yologarch\_(Earth-616)

NaN

Bad

Characters

NaN

Na

1093 rows × 11 columns

|  |  |  |  |
| --- | --- | --- | --- |
|  | | son) (Earth-616) | 616) Identity Characters Character |
| **16373** | 695217 | Tinkerer (Skrull) | \/Tinkerer\_(Skrull)\_(Earth-616) Secret Bad Bald Mal  Identity Characters Character |
|  |  | (Earth-616) |  |
| **16374** | 708811 | TK421  (Spiderling) | \/TK421\_(Spiderling)\_(Earth-616) Secret Neutral NaN Mal  Identity Characters Character |
|  |  | (Earth-616) |  |

In [158]:

*# фильтр по колонке*

data\_num[data\_num.index.isin(flt\_index)]['APPEARANCES']

Out[158]: 15280 NaN

15281 NaN

15282 NaN

15283 NaN

15284 NaN

..

16371 NaN

16372 NaN

16373 NaN

16374 NaN

16375 NaN

Name: APPEARANCES, Length: 1093, dtype: float64

In [159]:

data\_num\_APPEARANCES = data\_num[['APPEARANCES']] data\_num\_APPEARANCES.head()

|  |  |  |
| --- | --- | --- |
| Out[159]: |  | |
|  |  | **APPEARANCES** |
|  | **0** | 4043.0 |
|  | **1** | 3360.0 |
|  | **2** | 3061.0 |
|  | **3** | 2961.0 |
|  | **4** | 2258.0 |

In [160]:

**from sklearn.impute import** SimpleImputer

**from sklearn.impute import** MissingIndicator

In [161]:

*# Фильтр для проверки заполнения пустых значений*

indicator = MissingIndicator()

mask\_missing\_values\_only = indicator.fit\_transform(data\_num\_APPEARANCES) mask\_missing\_values\_only

Out[161]: array([[False],

[False],

[False],

...,

[ True],

[ True],

[ True]])

In [162]:

strategies=['mean', 'median','most\_frequent']

In [163]:

**def** test\_num\_impute(strategy\_param):

imp\_num = SimpleImputer(strategy=strategy\_param)

data\_num\_imp = imp\_num.fit\_transform(data\_num\_APPEARANCES)

**return** data\_num\_imp[mask\_missing\_values\_only]

In [164]:

strategies[0], test\_num\_impute(strategies[0])

Out[164]: ('mean',

array([17.03337696, 17.03337696, 17.03337696, ..., 17.03337696,

17.03337696, 17.03337696]))

In [165]:

strategies[1], test\_num\_impute(strategies[1])

Out[165]: ('median', array([3., 3., 3., ..., 3., 3., 3.]))

In [166]:

strategies[2], test\_num\_impute(strategies[2])

Out[166]: ('most\_frequent', array([1., 1., 1., ..., 1., 1., 1.]))

Заменим все пустые данные столбца 'APPEARANCES' на данные

test\_num\_impute(strategies[2])

In [167]:

new\_APPEARANCES = pd.DataFrame({'id': flt\_index, 'APPEARANCES':test\_num\_impute(strategies[2])})

new\_APPEARANCES

|  |  |  |  |
| --- | --- | --- | --- |
| Out[167]: |  | | |
|  |  | **id** | **APPEARANCES** |
|  | **0** | 15280 | 1.0 |
|  | **1** | 15281 | 1.0 |
|  | **2** | 15282 | 1.0 |
|  | **3** | 15283 | 1.0 |
|  | **4** | 15284 | 1.0 |
|  | **...** | ... | ... |
|  | **1088** | 16371 | 1.0 |
|  | **1089** | 16372 | 1.0 |
|  | **1090** | 16373 | 1.0 |
|  | **1091** | 16374 | 1.0 |
|  | **1092** | 16375 | 1.0 |

1093 rows × 2 columns

In [168]:

**for** index, row **in** new\_APPEARANCES.iterrows(): data.loc[row['id'], 'APPEARANCES'] = row['APPEARANCES']

data

Out[168]:

**page\_id name urlslug ID ALIGN HAIR**

**0**

Spider- 1678 Man (Peter

Parker)

\/Spider-Man\_(Peter\_Parker)

Secret

Good Brown

Identity Characters

Hair C

**1** 7139

Captain America (Steven Rogers)

\/Captain\_America\_(Steven\_Rogers) Public

Identity

Good Characters

White Hair C

**2** 64786

Wolverine (James

\"Logan\"

Howlett)

\/Wolverine\_(James\_%22Logan%22\_Howlett) Public Neutral

Identity Characters

Black

Hair C

**3** 1868

Iron Man (Anthony

\"Tony\" Stark)

\/Iron\_Man\_(Anthony\_%22Tony%22\_Stark) Public

Identity

Good Characters

Black Hair C

Identity

|  |  |  |
| --- | --- | --- |
| Thor (Thor No Good  **4** 2460 Odinson) \/Thor\_(Thor\_Odinson) Dual Characters | Blond Hair | C |
| **...** ... ... ... ... ... | ... |  |
| Ru'ach No Bad  **16371** 657508 (Earth-616) \/Ru%27ach\_(Earth-616) Dual Characters | No Hair | C |

**16372** 665474

Thane (Thanos'

\/Thane\_(Thanos%27\_son)\_(Earth-616)

Identity

No Dual

|  |  |  |  |
| --- | --- | --- | --- |
|  | | son) (Earth-616) | Identity Characters C |
| **16373** | 695217 | Tinkerer (Skrull) | \/Tinkerer\_(Skrull)\_(Earth-616) Secret Bad Bald  Identity Characters C |
|  |  | (Earth-616) |  |
| **16374** | 708811 | TK421  (Spiderling) | \/TK421\_(Spiderling)\_(Earth-616) Secret Neutral NaN  Identity Characters C |
|  |  | (Earth-616) |  |

Good

Bald

**16375** 673702

Yologarch

(Earth-616)

\/Yologarch\_(Earth-616)

NaN

Bad

Characters

NaN

16373 rows × 11 columns

В столбце 'APPEARANCES' больше нет пропущенных данных:

In [169]:

data.isnull().sum()

Out[169]: page\_id 0

name 0

urlslug 0

ID 3767

ALIGN 2809

HAIR 4261

SEX 851

ALIVE 0

APPEARANCES 0

FIRST APPEARANCE 812

Year 812

dtype: int64

### Обработка пропусков в категориальных данных

In [170]:

*# 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\_n ull\_count, temp\_perc))

Колонка ID. Тип данных object. Количество пустых значений 3767, 23.0%. Колонка ALIGN. Тип данных object. Количество пустых значений 2809, 17.15%. Колонка HAIR. Тип данных object. Количество пустых значений 4261, 26.02%. Колонка SEX. Тип данных object. Количество пустых значений 851, 5.2%.

Колонка FIRST APPEARANCE. Тип данных object. Количество пустых значений 812, 4.96%.

In [171]:

cat\_temp\_data = data[['SEX']] cat\_temp\_data.head()

Out[171]:

**SEX**

**0** Male Characters

**1** Male Characters

**2** Male Characters

**3** Male Characters

**4** Male Characters

In [172]:

cat\_temp\_data['SEX'].unique()

Out[172]: array(['Male Characters', 'Female Characters', 'Genderfluid Characters', 'Agender Characters', nan], dtype=object)

In [173]:

cat\_temp\_data[cat\_temp\_data['SEX'].isnull()].shape

Out[173]: (851, 1)

In [174]:

*# Импьютация наиболее частыми значениями*

imp2 = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent') data\_imp2 = imp2.fit\_transform(cat\_temp\_data)

data\_imp2

Out[174]: array([['Male Characters'],

['Male Characters'], ['Male Characters'],

...,

['Male Characters'], ['Male Characters'],

['Male Characters']], dtype=object)

In [175]:

*# Пустые значения отсутствуют*

np.unique(data\_imp2)

Out[175]: array(['Agender Characters', 'Female Characters',

'Genderfluid Characters', 'Male Characters'], dtype=object)

# Преобразование категориальных признаков в числовые

In [176]:

cat\_enc = pd.DataFrame({'c1':data\_imp2.T[0]}) cat\_enc

|  |  |  |
| --- | --- | --- |
| Out[176]: |  | |
|  |  | **c1** |
|  | **0** | Male Characters |
|  | **1** | Male Characters |
|  | **2** | Male Characters |
|  | **3** | Male Characters |
|  | **4** | Male Characters |
|  | **...** | ... |
|  | **16368** | Male Characters |
|  | **16369** | Male Characters |
|  | **16370** | Male Characters |
|  | **16371** | Male Characters |
|  | **16372** | Male Characters |

16373 rows × 1 columns

## Кодирование категорий целочисленными значениями

**- label encoding**

In [177]:

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

In [178]:

le = LabelEncoder()

cat\_enc\_le = le.fit\_transform(cat\_enc['c1'])

In [179]:

cat\_enc['c1'].unique()

Out[179]: array(['Male Characters', 'Female Characters', 'Genderfluid Characters', 'Agender Characters'], dtype=object)

In [180]:

np.unique(cat\_enc\_le)

Out[180]: array([0, 1, 2, 3])

In [181]:

le.inverse\_transform([0, 1, 2, 3])

Out[181]: array(['Agender Characters', 'Female Characters',

'Genderfluid Characters', 'Male Characters'], dtype=object)

# Масштабирование данных

In [184]:

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

In [188]:

sc1 = MinMaxScaler()

sc1\_data = sc1.fit\_transform(data[['Year']])

In [189]:

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

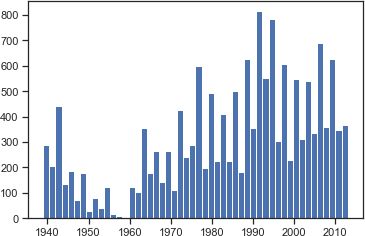
c:\users\user\appdata\local\programs\python\python37-32\lib\site-package s\numpy\lib\histograms.py:839: RuntimeWarning: invalid value encountered

in greater\_equal

keep = (tmp\_a >= first\_edge) c:\users\user\appdata\local\programs\python\python37-32\lib\site-package s\numpy\lib\histograms.py:840: RuntimeWarning: invalid value encountered

in less\_equal

keep &= (tmp\_a <= last\_edge)



In [190]:

plt.hist(sc1\_data, 50) plt.show()

