[Имя автора]

[Адрес электронной почты]

Аннотация

[Заинтересуйте читателя с помощью аннотации (как правило, это краткое содержание документа).   
Если вы готовы добавить свой текст — просто щелкните здесь и введите его.]

Lab3

[Подзаголовок документа]

*Государственное образовательное учреждение высшего профессионального образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

**«Лабораторная работа №3»**

«Технология машинного обучения»

**ИСПОЛНИТЕЛЬ:**

Студен группы РТ5-61

Курьянов А.И.

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е.

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2020 г.

Лабораторная работа №3

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.) Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи: 1 - обработку пропусков в данных; 2 - кодирование категориальных признаков; 3 - масштабирование данных.

Датасет: <https://www.kaggle.com/fivethirtyeight/fivethirtyeight-comic-characters-dataset>

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 [5]:

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

In [6]:

data.shape

Out[6]:

(6896, 13)

In [7]:

data.isnull().sum()

|  |  |
| --- | --- |
| Out[7]:  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

In [8]:

data.head()

Out[8]:

**page\_id name urlslug ID ALIGN EYE HAIR SEX GSM ALIVE APPEARANCES FIR**

**APPEARAN**

**1** 23387

Superman

(Clark Kent)

\/wiki\/Superman\_(Clark\_Kent)

Secret Identity

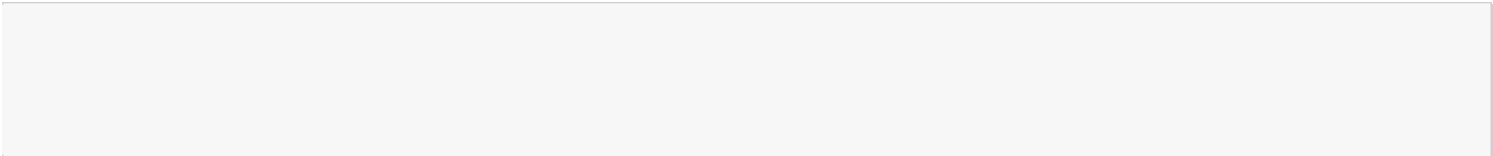
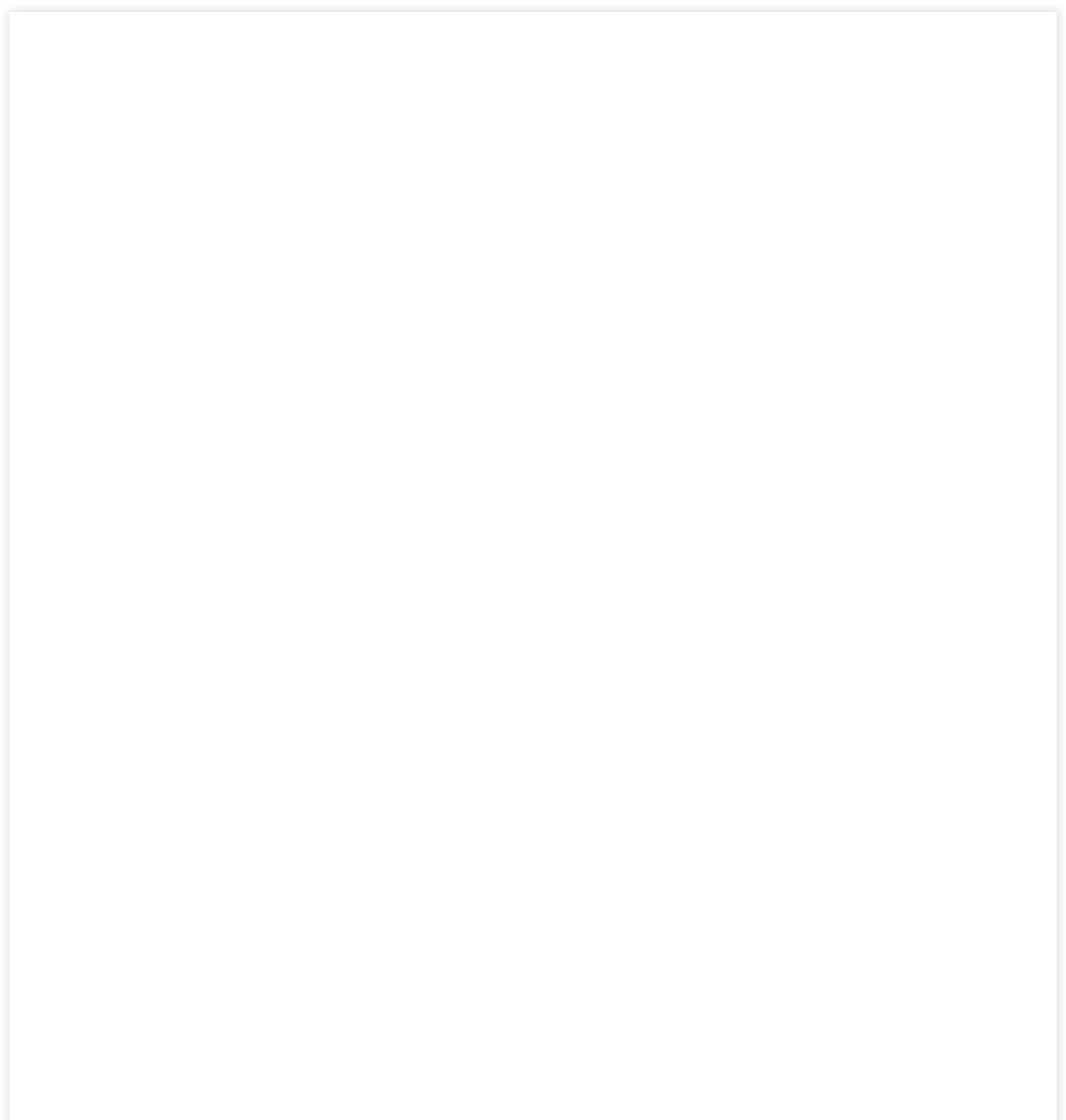
Good Characters

Blue Eyes

Black Hair

Male NaN Characters

Living 2496.0 1986, Octo Characters



**0** 1422

Batman

(Bruce Wayne)

\/wiki\/Batman\_(Bruce\_Wayne)

Secret

Good Blue Black

Male

Identity Characters Eyes

Hair Characters

NaN

Living

Characters

3093.0

1939, M

**2** 1458

Green

Lantern

(Hal Jordan)

\/wiki\/Green\_Lantern\_(Hal\_Jordan)

Secret

Good Brown Brown

Male

Identity Characters Eyes

Hair Characters

NaN

Living

Characters

1565.0 1959, Octo

**3** 1659

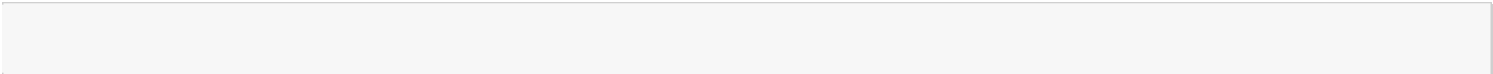
James Gordon (New Earth)

\/wiki\/James\_Gordon\_(New\_Earth) Public

Identity

Good Characters

In [10]:



total\_count = data.shape[0]

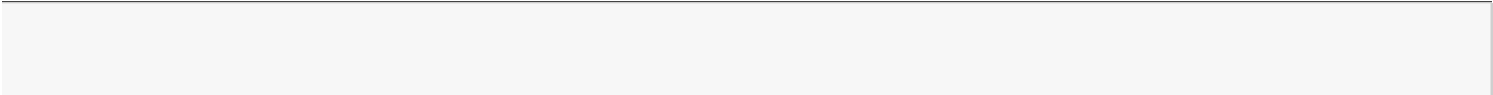
print('Всего строк: **{}**'.format(total\_count))



Всего строк: 6896

1. **Обработка пропусков в данных**

In [11]:

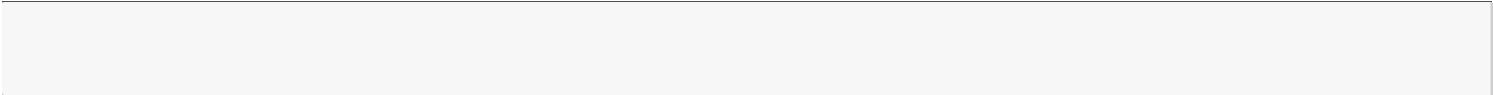


*# Удаление колонок, содержащих пустые значения* data\_new\_1 = data.dropna(axis=1, how='any') (data.shape, data\_new\_1.shape)

Out[11]:

((6896, 13), (6896, 3))

In [12]:



*# Удаление строк, содержащих пустые значения* data\_new\_2 = data.dropna(axis=0, how='any') (data.shape, data\_new\_2.shape)

Out[12]:

((6896, 13), (38, 13))

In [13]:



data.head()

Out[13]:

**page\_id name urlslug ID ALIGN EYE HAIR SEX GSM ALIVE APPEARANCES FIR**

**APPEARAN**

**0** 1422

Batman

(Bruce Wayne)

\/wiki\/Batman\_(Bruce\_Wayne)

Secret

Good Blue Black

Male

Identity Characters Eyes

Hair Characters

NaN

Living

Characters

3093.0

1939, M

**1** 23387

Superman

(Clark Kent)

\/wiki\/Superman\_(Clark\_Kent)

Secret Identity

Good Characters

Blue Eyes

Black Hair

Male NaN Characters

Living 2496.0 1986, Octo Characters

**2** 1458

Green

Lantern

(Hal Jordan)

\/wiki\/Green\_Lantern\_(Hal\_Jordan)

Secret

Good Brown Brown

Male

Identity Characters Eyes

Hair Characters

NaN

Living

Characters

1565.0 1959, Octo

**3** 1659

James Gordon (New Earth)

\/wiki\/James\_Gordon\_(New\_Earth) Public

Identity

Good Characters

Brown Eyes

White Hair

Male Characters

NaN Living Characters

1316.0 1987, Febru



Batman

Secret

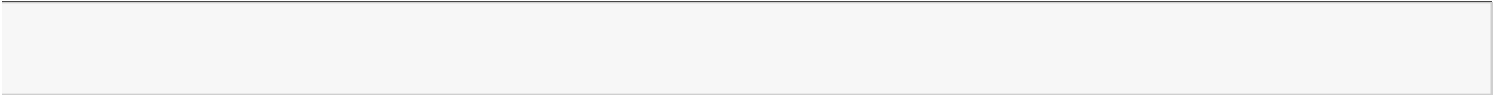
Good Blue Black

Male

Living

|  |  |  |
| --- | --- | --- |
| Richard  **4** 1576 Grayson \/wiki\/Richard\_Grayson\_(New\_Earth) Secret Good Blue Black Male NaN Living 1237.0 1940, A (New Identity Characters Eyes Hair Characters Characters  Earth) | | |
|  |  |  |

In [14]:



*# Заполнение всех пропущенных значений нулями*

data\_new\_3 = data.fillna(0) data\_new\_3.head()



Out[14]:

**page\_id name urlslug ID ALIGN EYE HAIR SEX GSM ALIVE APPEARANCES FIR**

**APPEARAN**

**0** 1422

**page\_id**

(Bruce

W**n**a**a**yn**m**e**e**)

\/wiki\/Batman\_(Bruce\_Wayne)

**urlslug ID ALIGN EYE HAIR**

Identity Characters Eyes Hair Characters

0

**SEX GSM ALIVE APPEARANCES**

Characters

3093.0 1939, M

**FIR**

**APPEARAN**

**1** 23387

Superman

(Clark Kent)

\/wiki\/Superman\_(Clark\_Kent)

Secret Identity

Good Characters

Blue Eyes

Black Hair

Male 0

Characters

Living 2496.0 1986, Octo Characters

**2** 1458

Green

Lantern

(Hal Jordan)

\/wiki\/Green\_Lantern\_(Hal\_Jordan)

Secret

Good Brown Brown

Male

Identity Characters Eyes

Hair Characters

0

Living

Characters

1565.0 1959, Octo

**3** 1659

James Gordon (New Earth)

\/wiki\/James\_Gordon\_(New\_Earth) Public

Identity

Good Characters

Brown Eyes

White Hair

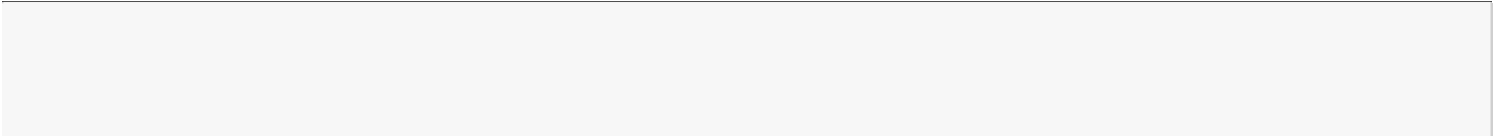
Male Characters

0 Living

Characters

1316.0 1987, Febru





*# Гистограмма по признакам*

**for** col **in** data\_num:

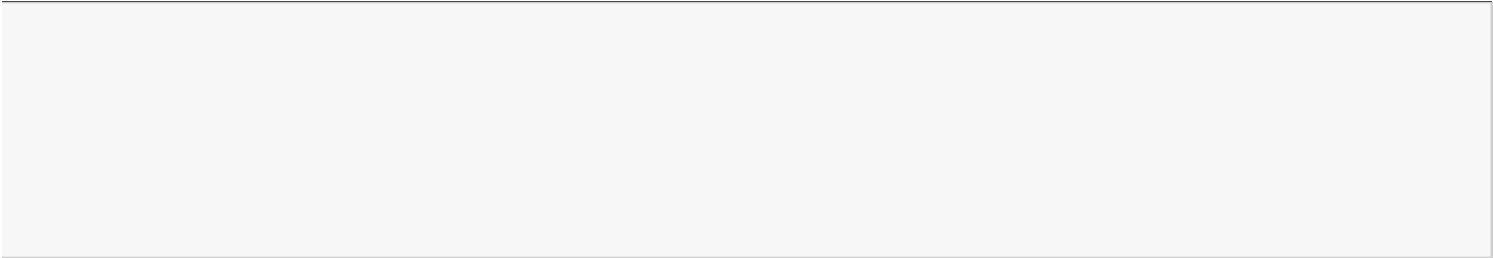
plt.hist(data[col], 50) plt.xlabel(col) plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\histograms.py:839: RuntimeWarning: invalid value encountered in greater\_equal keep = (tmp\_a >= first\_edge)

C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\histograms.py:840: RuntimeWarning: invalid value encountered in less\_equal keep &= (tmp\_a <= last\_edge)

|  |  |  |
| --- | --- | --- |
| Richard  **4** 1576 Grayson \/wiki\/Richard\_Grayson\_(New\_Earth) Secret Good Blue Black Male 0 Living 1237.0 1940, A (New Identity Characters Eyes Hair Characters Characters  Earth) | | |
|  |  |  |

In [15]:



*# 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)

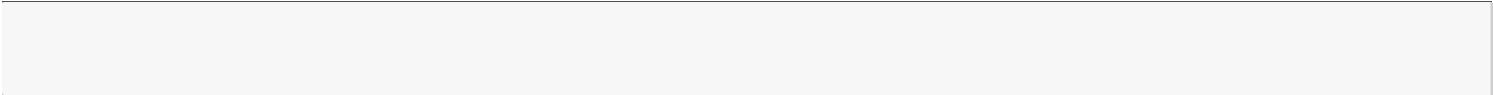
temp\_perc = round((temp\_null\_count / total\_count) \* 100.0, 2)

print('Колонка **{}**. Тип данных **{}**. Количество пустых значений **{}**, **{}**%.'.format(col, dt, temp\_null\_count, temp\_perc))



Колонка APPEARANCES. Тип данных float64. Количество пустых значений 355, 5.15%. Колонка YEAR. Тип данных float64. Количество пустых значений 69, 1.0%.

In [16]:



*# Фильтр по колонкам с пропущенными значениями*

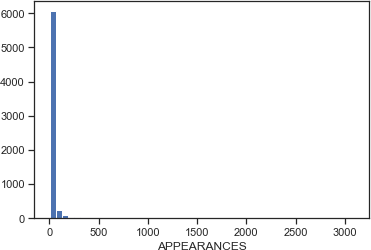
data\_num = data[num\_cols] data\_num

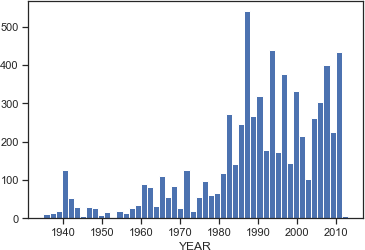
Out[16]:

|  |  |  |
| --- | --- | --- |
|  | **APPEARANCES** | **YEAR** |
| **0** | 3093.0 | 1939.0 |
| **1** | 2496.0 | 1986.0 |
| **2** | 1565.0 | 1959.0 |
| **3** | 1316.0 | 1987.0 |
| **4** | 1237.0 | 1940.0 |
| **...** | ... | ... |
| **6891** | NaN | NaN |
| **6892** | NaN | NaN |
| **6893** | NaN | NaN |
| **6894** | NaN | NaN |
| **6895** | NaN | NaN |

6896 rows × 2 columns

In [17]:





In [18]:



*# Фильтр по пустым значениям поля APPEARANCES*

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

Out[18]:

**page\_id name urlslug ID ALIGN EYE HAIR SEX GSM ALIVE APPEARANCES APPEA**

Matteo

**6541** 306472 Bischoff (New Earth)

\/wiki\/Matteo\_Bischoff\_(New\_Earth)

Secret

Bad

Identity Characters

NaN

Grey

Male

Hair Characters

NaN

Living

Characters

NaN

2

**6542** 273317 Doomslayer

(New Earth)

**6543** 242097

Emily Sung

(New Earth)

\/wiki\/Emily\_Sung\_(New\_Earth)

Secret

Good Violet Purple Female

Identity Characters Eyes

Hair Characters

NaN

Living

Characters

NaN

2

\/wiki\/Doomslayer\_(New\_Earth) Secret

Identity

Bad Characters

Green Eyes

White Hair

Male Characters

NaN Living Characters

NaN 2

**6544** 247494 Ry'jll (New

Earth)

Baron

**6545** 161599 Gestapo

(New Earth)

\/wiki\/Baron\_Gestapo\_(New\_Earth)

NaN

Bad

Characters

NaN NaN

Male

Characters

NaN

Living

Characters

NaN 201

\/wiki\/Ry%27jll\_(New\_Earth) Secret

Identity

Good Characters

Green Eyes

NaN Female Characters

NaN Living Characters

NaN 2011,

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

**6891**

Nadine

66302 West (New

Earth)

\/wiki\/Nadine\_West\_(New\_Earth)

Public

Good

Identity Characters

NaN NaN

Female

Characters

NaN

Living

Characters

NaN

**6892** 283475

Warren Harding (New Earth)

\/wiki\/Warren\_Harding\_(New\_Earth)

Public Identity

Good NaN NaN Characters

Male NaN Characters

Living NaN

Characters

**6894** 283471

William McKinley (New Earth)

\/wiki\/William\_McKinley\_(New\_Earth)

Public Identity

Good NaN NaN Characters

Male NaN Characters

Living NaN

Characters

355 rows × 13 columns



*# Запоминаем индексы строк с пустыми значениями*

William

**6893** 283478 Harrison \/wiki\/William\_Harrison\_(New\_Earth) (New Earth)

Public

Good

Identity Characters

NaN NaN

Male

Characters

NaN

Living

Characters

NaN

**6895** 150660

Mookie

(New Earth)

\/wiki\/Mookie\_(New\_Earth)

Public

Bad Blue Blond

Male

Identity Characters Eyes

Hair Characters

NaN

Living

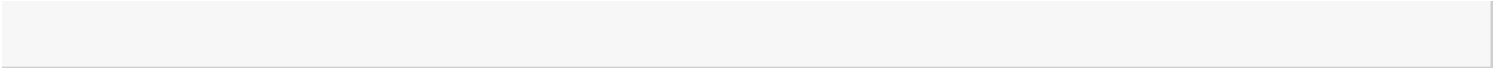
Characters

NaN



|  |  |  |
| --- | --- | --- |
|  |  |  |

In [19]:



flt\_index = data[data['APPEARANCES'].isnull()].index flt\_index

Out[19]:

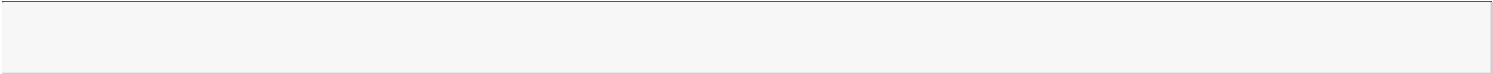
Int64Index([6541, 6542, 6543, 6544, 6545, 6546, 6547, 6548, 6549, 6550,

...

6886, 6887, 6888, 6889, 6890, 6891, 6892, 6893, 6894, 6895],

dtype='int64', length=355)

In [20]:



*# Проверяем что выводятся нужные строки*

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

Out[20]:

**page\_id name urlslug ID ALIGN EYE HAIR SEX GSM ALIVE APPEARANCES APPEA**

Matteo

**6541** 306472 Bischoff (New Earth)

\/wiki\/Matteo\_Bischoff\_(New\_Earth)

Secret

Bad

Identity Characters

NaN

Grey

Male

Hair Characters

NaN

Living

Characters

NaN

2

**6542** 273317 Doomslayer

(New Earth)

**6543** 242097

Emily Sung

(New Earth)

\/wiki\/Emily\_Sung\_(New\_Earth)

Secret

Good Violet Purple Female

Identity Characters Eyes

Hair Characters

NaN

Living

Characters

NaN

2

\/wiki\/Doomslayer\_(New\_Earth) Secret

Identity

Bad Characters

Green Eyes

White Hair

Male Characters

NaN Living Characters

NaN 2

**6544** 247494 Ry'jll (New

Earth)

Baron

**6545** 161599 Gestapo

(New Earth)

\/wiki\/Baron\_Gestapo\_(New\_Earth)

NaN

Bad

Characters

NaN NaN

Male

Characters

NaN

Living

Characters

NaN 201

\/wiki\/Ry%27jll\_(New\_Earth) Secret

Identity

Good Characters

Green Eyes

NaN Female Characters

NaN Living Characters

NaN 2011,

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

**6891**

Nadine

66302 West (New

Earth)

\/wiki\/Nadine\_West\_(New\_Earth)

Public

Good

Identity Characters

NaN NaN

Female

Characters

NaN

Living

Characters

NaN

**6892** 283475

Warren Harding (New Earth)

\/wiki\/Warren\_Harding\_(New\_Earth)

Public Identity

Good NaN NaN Characters

Male NaN Characters

Living NaN

Characters

**6894** 283471

William McKinley (New Earth)

\/wiki\/William\_McKinley\_(New\_Earth)

Public Identity

Good NaN NaN Characters

Male NaN Characters

Living NaN

Characters

355 rows × 13 columns

William

**6893** 283478 Harrison \/wiki\/William\_Harrison\_(New\_Earth) (New Earth)

Public

Good

Identity Characters

NaN NaN

Male

Characters

NaN

Living

Characters

NaN

**6895** 150660

Mookie

(New Earth)

\/wiki\/Mookie\_(New\_Earth)

Public

Bad Blue Blond

Male

Identity Characters Eyes

Hair Characters

NaN

Living

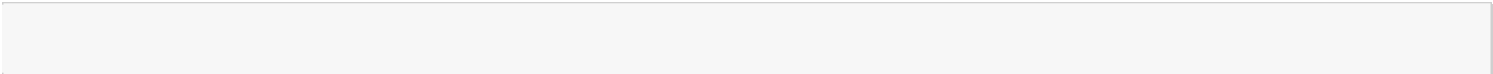
Characters

NaN



|  |  |  |
| --- | --- | --- |
|  |  |  |

In [21]:



*# фильтр по колонке*

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

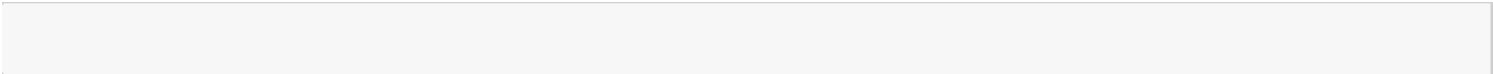
Out[21]:

.

|  |  |
| --- | --- |
| 6541 | NaN |
| 6542 | NaN |
| 6543 | NaN |
| 6544 | NaN |
| 6545 | NaN |
| . |  |
| 6891 | NaN |
| 6892 | NaN |
| 6893 | NaN |
| 6894 | NaN |
| 6895 | NaN |

Name: APPEARANCES, Length: 355, dtype: float64

In [22]:



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

Out[22]:

**APPEARANCES**

**0** 3093.0

**1** 2496.0

**2** 1565.0

**3** 1316.0

**4** 1237.0

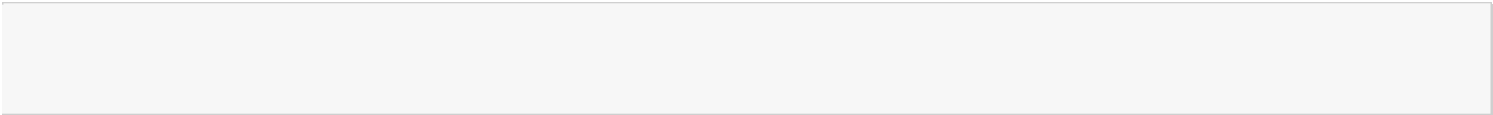
In [23]:



**from sklearn.impute import** SimpleImputer

**from sklearn.impute import** MissingIndicator

In [24]:



*# Фильтр для проверки заполнения пустых значений*

indicator = MissingIndicator()

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

Out[24]:

array([[False], [False],

[False],

...,

[ True],

[ True],

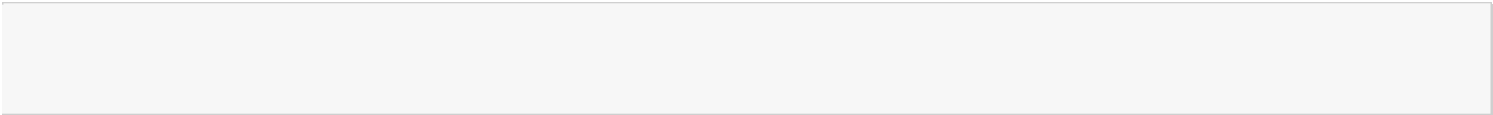
[ True]])

In [28]:



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

In [29]:



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

imp\_num = SimpleImputer(strategy=strategy\_param) data\_num\_imp = imp\_num.fit\_transform(data\_num\_MasVnrArea) **return** data\_num\_imp[mask\_missing\_values\_only]

In [30]:



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

Out[30]:

('mean',

array([23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,



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

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377,

23.62513377, 23.62513377, 23.62513377, 23.62513377, 23.62513377]))

In [31]:



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

Out[31]:

('median',

array([6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.,

6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6., 6.]))

In [32]:



Out[32]:

('most\_frequent',

array([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., 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., 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., 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., 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., 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., 1.,

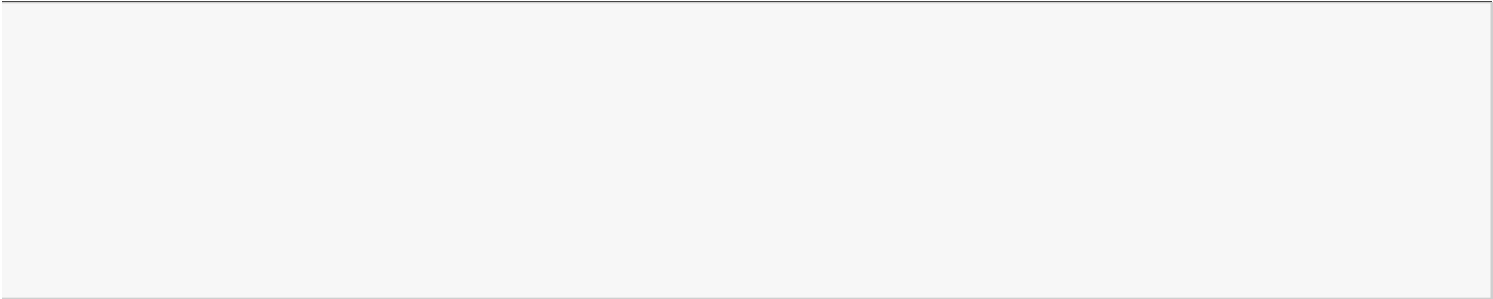
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,

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., 1., 1., 1., 1.,

1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]))

In [33]:



*# 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\_data[filled\_data.size-1]

In [35]:



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

|  |  |
| --- | --- |
| Out[35]: |  |
|  | **APPEARANCES** |
| **count** | 6541.000000 |
| **mean** | 23.625134 |
| **std** | 87.378509 |
| **min** | 1.000000 |
| **25%** | 2.000000 |
| **50%** | 6.000000 |
| **75%** | 15.000000 |
| **max** | 3093.000000 |

In [36]:



test\_num\_impute\_col(data, 'APPEARANCES', strategies[0])

Out[36]:

('APPEARANCES', 'mean', 355, 23.62513377159456, 23.62513377159456)

In [37]:



test\_num\_impute\_col(data, 'APPEARANCES', strategies[1])

Out[37]:

('APPEARANCES', 'median', 355, 6.0, 6.0)



In [38]:

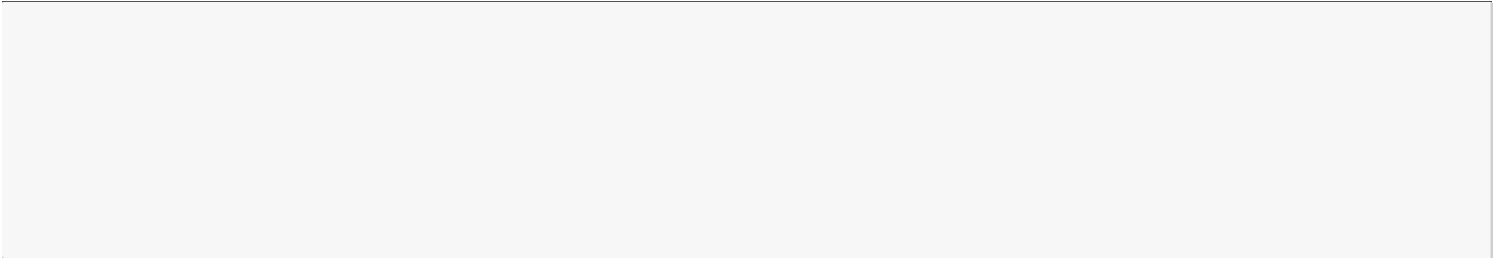


test\_num\_impute\_col(data, 'APPEARANCES', strategies[2])

Out[38]:

('APPEARANCES', 'most\_frequent', 355, 1.0, 1.0)

In [39]:



*# 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 ID. Тип данных object. Количество пустых значений 2013, 29.19%. Колонка ALIGN. Тип данных object. Количество пустых значений 601, 8.72%. Колонка EYE. Тип данных object. Количество пустых значений 3628, 52.61%. Колонка HAIR. Тип данных object. Количество пустых значений 2274, 32.98%. Колонка SEX. Тип данных object. Количество пустых значений 125, 1.81%.

Колонка GSM. Тип данных object. Количество пустых значений 6832, 99.07%. Колонка ALIVE. Тип данных object. Количество пустых значений 3, 0.04%.

Колонка FIRST APPEARANCE. Тип данных object. Количество пустых значений 69, 1.0%.

In [40]:



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

Out[40]:

**APPEARANCES**

**0** 3093.0

**1** 2496.0

**2** 1565.0

**3** 1316.0

**4** 1237.0

In [41]:



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

Out[41]:

array([3.093e+03, 2.496e+03, 1.565e+03, 1.316e+03, 1.237e+03, 1.231e+03, 1.121e+03, 1.095e+03, 1.075e+03, 1.028e+03, 9.690e+02, 9.510e+02,

9.340e+02, 9.300e+02, 8.030e+02, 7.160e+02, 7.060e+02, 6.770e+02,

6.540e+02, 6.350e+02, 6.050e+02, 5.950e+02, 5.930e+02, 5.840e+02,

5.600e+02, 5.580e+02, 5.570e+02, 5.490e+02, 5.170e+02, 4.920e+02,

4.870e+02, 4.700e+02, 4.390e+02, 4.360e+02, 4.290e+02, 4.270e+02,

4.230e+02, 4.220e+02, 4.130e+02, 3.990e+02, 3.930e+02, 3.910e+02,

3.880e+02, 3.860e+02, 3.740e+02, 3.710e+02, 3.700e+02, 3.610e+02,

3.560e+02, 3.530e+02, 3.500e+02, 3.450e+02, 3.440e+02, 3.360e+02,

3.350e+02, 3.250e+02, 3.210e+02, 3.190e+02, 3.140e+02, 3.110e+02,

3.090e+02, 3.080e+02, 3.060e+02, 3.050e+02, 3.010e+02, 3.000e+02,

2.990e+02, 2.970e+02, 2.910e+02, 2.880e+02, 2.860e+02, 2.840e+02,

2.820e+02, 2.680e+02, 2.660e+02, 2.620e+02, 2.610e+02, 2.560e+02,

2.540e+02, 2.530e+02, 2.520e+02, 2.500e+02, 2.450e+02, 2.410e+02,

2.390e+02, 2.370e+02, 2.350e+02, 2.340e+02, 2.310e+02, 2.290e+02,

2.280e+02, 2.270e+02, 2.260e+02, 2.250e+02, 2.240e+02, 2.220e+02,

2.190e+02, 2.180e+02, 2.160e+02, 2.140e+02, 2.130e+02, 2.120e+02,

2.080e+02, 2.070e+02, 2.060e+02, 1.990e+02, 1.980e+02, 1.920e+02,

1.910e+02, 1.890e+02, 1.880e+02, 1.840e+02, 1.830e+02, 1.820e+02,

1.810e+02, 1.800e+02, 1.790e+02, 1.770e+02, 1.760e+02, 1.750e+02,

1.740e+02, 1.730e+02, 1.710e+02, 1.700e+02, 1.680e+02, 1.670e+02,

1.660e+02, 1.650e+02, 1.640e+02, 1.630e+02, 1.620e+02, 1.610e+02,

1.600e+02, 1.590e+02, 1.580e+02, 1.570e+02, 1.550e+02, 1.540e+02,

1.530e+02, 1.520e+02, 1.510e+02, 1.480e+02, 1.470e+02, 1.430e+02,

1.420e+02, 1.410e+02, 1.400e+02, 1.390e+02, 1.380e+02, 1.370e+02,

1.360e+02, 1.340e+02, 1.330e+02, 1.320e+02, 1.310e+02, 1.300e+02,

1.290e+02, 1.280e+02, 1.270e+02, 1.260e+02, 1.250e+02, 1.240e+02,

1.230e+02, 1.220e+02, 1.210e+02, 1.200e+02, 1.180e+02, 1.160e+02,

1.150e+02, 1.140e+02, 1.130e+02, 1.110e+02, 1.100e+02, 1.090e+02,

1.080e+02, 1.070e+02, 1.060e+02, 1.050e+02, 1.040e+02, 1.030e+02,

1.020e+02, 1.010e+02, 1.000e+02, 9.900e+01, 9.800e+01, 9.700e+01,

9.600e+01, 9.500e+01, 9.400e+01, 9.300e+01, 9.200e+01, 9.100e+01,

9.000e+01, 8.900e+01, 8.800e+01, 8.700e+01, 8.600e+01, 8.500e+01,

8.400e+01, 8.300e+01, 8.200e+01, 8.100e+01, 8.000e+01, 7.900e+01,

7.800e+01, 7.700e+01, 7.600e+01, 7.500e+01, 7.400e+01, 7.300e+01,

7.200e+01, 7.100e+01, 7.000e+01, 6.900e+01, 6.800e+01, 6.700e+01,

6.600e+01, 6.500e+01, 6.400e+01, 6.300e+01, 6.200e+01, 6.100e+01,

6.000e+01, 5.900e+01, 5.800e+01, 5.700e+01, 5.600e+01, 5.500e+01,

5.400e+01, 5.300e+01, 5.200e+01, 5.100e+01, 5.000e+01, 4.900e+01,

4.800e+01, 4.700e+01, 4.600e+01, 4.500e+01, 4.400e+01, 4.300e+01,

4.200e+01, 4.100e+01, 4.000e+01, 3.900e+01, 3.800e+01, 3.700e+01,

3.600e+01, 3.500e+01, 3.400e+01, 3.300e+01, 3.200e+01, 3.100e+01,

3.000e+01, 2.900e+01, 2.800e+01, 2.700e+01, 2.600e+01, 2.500e+01,

2.400e+01, 2.300e+01, 2.200e+01, 2.100e+01, 2.000e+01, 1.900e+01,

1.800e+01, 1.700e+01, 1.600e+01, 1.500e+01, 1.400e+01, 1.300e+01,

1.200e+01, 1.100e+01, 1.000e+01, 9.000e+00, 8.000e+00, 7.000e+00,

6.000e+00, 5.000e+00, 4.000e+00, 3.000e+00, 2.000e+00, 1.000e+00,

nan])

In [42]:

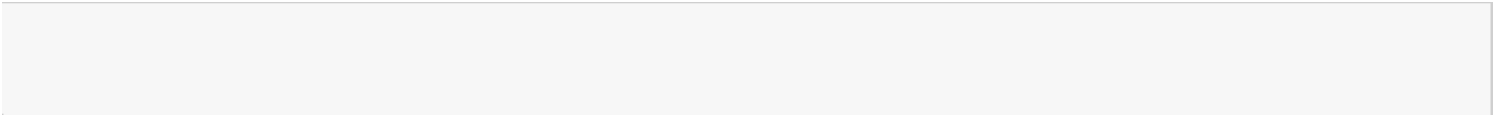


cat\_temp\_data[cat\_temp\_data['APPEARANCES'].isnull()].shape

Out[42]:

(355, 1)

In [43]:



*# Импьютация наиболее частыми значениями*

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

data\_imp2

Out[43]:

array([[3.093e+03], [2.496e+03],

[1.565e+03],

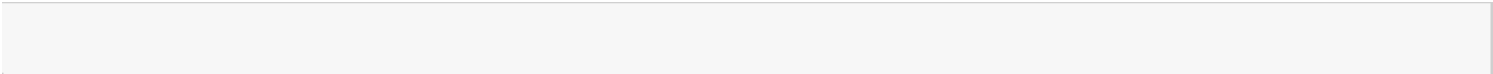
...,

[1.000e+00],

[1.000e+00],

[1.000e+00]])

In [44]:



*# Пустые значения отсутствуют*

np.unique(data\_imp2)

Out[44]:

array([1.000e+00, 2.000e+00, 3.000e+00, 4.000e+00, 5.000e+00, 6.000e+00, 7.000e+00, 8.000e+00, 9.000e+00, 1.000e+01, 1.100e+01, 1.200e+01,

1.300e+01, 1.400e+01, 1.500e+01, 1.600e+01, 1.700e+01, 1.800e+01,

1.900e+01, 2.000e+01, 2.100e+01, 2.200e+01, 2.300e+01, 2.400e+01,

2.500e+01, 2.600e+01, 2.700e+01, 2.800e+01, 2.900e+01, 3.000e+01,

3.100e+01, 3.200e+01, 3.300e+01, 3.400e+01, 3.500e+01, 3.600e+01,

3.700e+01, 3.800e+01, 3.900e+01, 4.000e+01, 4.100e+01, 4.200e+01,

4.300e+01, 4.400e+01, 4.500e+01, 4.600e+01, 4.700e+01, 4.800e+01,

4.900e+01, 5.000e+01, 5.100e+01, 5.200e+01, 5.300e+01, 5.400e+01,

5.500e+01, 5.600e+01, 5.700e+01, 5.800e+01, 5.900e+01, 6.000e+01,

6.100e+01, 6.200e+01, 6.300e+01, 6.400e+01, 6.500e+01, 6.600e+01,

6.700e+01, 6.800e+01, 6.900e+01, 7.000e+01, 7.100e+01, 7.200e+01,

7.300e+01, 7.400e+01, 7.500e+01, 7.600e+01, 7.700e+01, 7.800e+01,

7.900e+01, 8.000e+01, 8.100e+01, 8.200e+01, 8.300e+01, 8.400e+01,

8.500e+01, 8.600e+01, 8.700e+01, 8.800e+01, 8.900e+01, 9.000e+01,

9.100e+01, 9.200e+01, 9.300e+01, 9.400e+01, 9.500e+01, 9.600e+01,

9.700e+01, 9.800e+01, 9.900e+01, 1.000e+02, 1.010e+02, 1.020e+02,

1.030e+02, 1.040e+02, 1.050e+02, 1.060e+02, 1.070e+02, 1.080e+02,

1.090e+02, 1.100e+02, 1.110e+02, 1.130e+02, 1.140e+02, 1.150e+02,

1.160e+02, 1.180e+02, 1.200e+02, 1.210e+02, 1.220e+02, 1.230e+02,

1.240e+02, 1.250e+02, 1.260e+02, 1.270e+02, 1.280e+02, 1.290e+02,

1.300e+02, 1.310e+02, 1.320e+02, 1.330e+02, 1.340e+02, 1.360e+02,

1.370e+02, 1.380e+02, 1.390e+02, 1.400e+02, 1.410e+02, 1.420e+02,

1.430e+02, 1.470e+02, 1.480e+02, 1.510e+02, 1.520e+02, 1.530e+02,

1.540e+02, 1.550e+02, 1.570e+02, 1.580e+02, 1.590e+02, 1.600e+02,

1.610e+02, 1.620e+02, 1.630e+02, 1.640e+02, 1.650e+02, 1.660e+02,

1.670e+02, 1.680e+02, 1.700e+02, 1.710e+02, 1.730e+02, 1.740e+02,

1.750e+02, 1.760e+02, 1.770e+02, 1.790e+02, 1.800e+02, 1.810e+02,

1.820e+02, 1.830e+02, 1.840e+02, 1.880e+02, 1.890e+02, 1.910e+02,

1.920e+02, 1.980e+02, 1.990e+02, 2.060e+02, 2.070e+02, 2.080e+02,

2.120e+02, 2.130e+02, 2.140e+02, 2.160e+02, 2.180e+02, 2.190e+02,

2.220e+02, 2.240e+02, 2.250e+02, 2.260e+02, 2.270e+02, 2.280e+02,

2.290e+02, 2.310e+02, 2.340e+02, 2.350e+02, 2.370e+02, 2.390e+02,

2.410e+02, 2.450e+02, 2.500e+02, 2.520e+02, 2.530e+02, 2.540e+02,

2.560e+02, 2.610e+02, 2.620e+02, 2.660e+02, 2.680e+02, 2.820e+02,

2.840e+02, 2.860e+02, 2.880e+02, 2.910e+02, 2.970e+02, 2.990e+02,

3.000e+02, 3.010e+02, 3.050e+02, 3.060e+02, 3.080e+02, 3.090e+02,

3.110e+02, 3.140e+02, 3.190e+02, 3.210e+02, 3.250e+02, 3.350e+02,

3.360e+02, 3.440e+02, 3.450e+02, 3.500e+02, 3.530e+02, 3.560e+02,

3.610e+02, 3.700e+02, 3.710e+02, 3.740e+02, 3.860e+02, 3.880e+02,

3.910e+02, 3.930e+02, 3.990e+02, 4.130e+02, 4.220e+02, 4.230e+02,

4.270e+02, 4.290e+02, 4.360e+02, 4.390e+02, 4.700e+02, 4.870e+02,

4.920e+02, 5.170e+02, 5.490e+02, 5.570e+02, 5.580e+02, 5.600e+02,

5.840e+02, 5.930e+02, 5.950e+02, 6.050e+02, 6.350e+02, 6.540e+02,

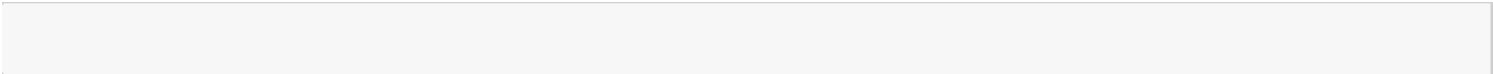
6.770e+02, 7.060e+02, 7.160e+02, 8.030e+02, 9.300e+02, 9.340e+02,

9.510e+02, 9.690e+02, 1.028e+03, 1.075e+03, 1.095e+03, 1.121e+03,

1.231e+03, 1.237e+03, 1.316e+03, 1.565e+03, 2.496e+03, 3.093e+03])

1. **Преобразование категориальных признаков в числовые**

In [46]:



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

Out[46]:

|  |  |
| --- | --- |
|  | **c1** |
| **0** | 3093.0 |
| **1** | 2496.0 |
| **2** | 1565.0 |
| **3** | 1316.0 |
| **4** | 1237.0 |
| **...** | ... |
| **6891** | 1.0 |
| **6892** | 1.0 |
| **6893** | 1.0 |
| **6894** | 1.0 |
| **6895** | 1.0 |

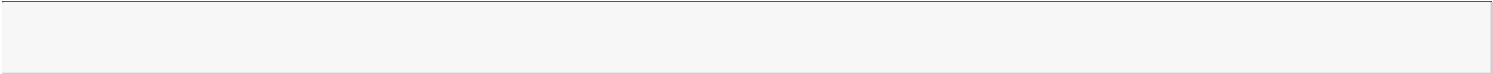
6896 rows × 1 columns

In [47]:



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

In [48]:



le = LabelEncoder()

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

In [49]:



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

Out[49]:

array([3.093e+03, 2.496e+03, 1.565e+03, 1.316e+03, 1.237e+03, 1.231e+03, 1.121e+03, 1.095e+03, 1.075e+03, 1.028e+03, 9.690e+02, 9.510e+02,

9.340e+02, 9.300e+02, 8.030e+02, 7.160e+02, 7.060e+02, 6.770e+0

 6.540e+02, 6.350e+02, 6.050e+02, 5.950e+02, 5.930e+02, 5.840e+02,

5.600e+02, 5.580e+02, 5.570e+02, 5.490e+02, 5.170e+02, 4.920e+02,

4.870e+02, 4.700e+02, 4.390e+02, 4.360e+02, 4.290e+02, 4.270e+02,

4.230e+02, 4.220e+02, 4.130e+02, 3.990e+02, 3.930e+02, 3.910e+02,

3.880e+02, 3.860e+02, 3.740e+02, 3.710e+02, 3.700e+02, 3.610e+02,

3.560e+02, 3.530e+02, 3.500e+02, 3.450e+02, 3.440e+02, 3.360e+02,

3.350e+02, 3.250e+02, 3.210e+02, 3.190e+02, 3.140e+02, 3.110e+02,

3.090e+02, 3.080e+02, 3.060e+02, 3.050e+02, 3.010e+02, 3.000e+02,

2.990e+02, 2.970e+02, 2.910e+02, 2.880e+02, 2.860e+02, 2.840e+02,

2.820e+02, 2.680e+02, 2.660e+02, 2.620e+02, 2.610e+02, 2.560e+02,

2.540e+02, 2.530e+02, 2.520e+02, 2.500e+02, 2.450e+02, 2.410e+02,

2.390e+02, 2.370e+02, 2.350e+02, 2.340e+02, 2.310e+02, 2.290e+02,

2.280e+02, 2.270e+02, 2.260e+02, 2.250e+02, 2.240e+02, 2.220e+02,

2.190e+02, 2.180e+02, 2.160e+02, 2.140e+02, 2.130e+02, 2.120e+02,

2.080e+02, 2.070e+02, 2.060e+02, 1.990e+02, 1.980e+02, 1.920e+02,

1.910e+02, 1.890e+02, 1.880e+02, 1.840e+02, 1.830e+02, 1.820e+02,

1.810e+02, 1.800e+02, 1.790e+02, 1.770e+02, 1.760e+02, 1.750e+02,

1.740e+02, 1.730e+02, 1.710e+02, 1.700e+02, 1.680e+02, 1.670e+02,

1.660e+02, 1.650e+02, 1.640e+02, 1.630e+02, 1.620e+02, 1.610e+02,

1.600e+02, 1.590e+02, 1.580e+02, 1.570e+02, 1.550e+02, 1.540e+02,

1.530e+02, 1.520e+02, 1.510e+02, 1.480e+02, 1.470e+02, 1.430e+02,

1.420e+02, 1.410e+02, 1.400e+02, 1.390e+02, 1.380e+02, 1.370e+02,

1.360e+02, 1.340e+02, 1.330e+02, 1.320e+02, 1.310e+02, 1.300e+02,

1.290e+02, 1.280e+02, 1.270e+02, 1.260e+02, 1.250e+02, 1.240e+02,

1.230e+02, 1.220e+02, 1.210e+02, 1.200e+02, 1.180e+02, 1.160e+02,

1.150e+02, 1.140e+02, 1.130e+02, 1.110e+02, 1.100e+02, 1.090e+02,

1.080e+02, 1.070e+02, 1.060e+02, 1.050e+02, 1.040e+02, 1.030e+02,

1.020e+02, 1.010e+02, 1.000e+02, 9.900e+01, 9.800e+01, 9.700e+01,

9.600e+01, 9.500e+01, 9.400e+01, 9.300e+01, 9.200e+01, 9.100e+01,

9.000e+01, 8.900e+01, 8.800e+01, 8.700e+01, 8.600e+01, 8.500e+01,

8.400e+01, 8.300e+01, 8.200e+01, 8.100e+01, 8.000e+01, 7.900e+01,

7.800e+01, 7.700e+01, 7.600e+01, 7.500e+01, 7.400e+01, 7.300e+01,

7.200e+01, 7.100e+01, 7.000e+01, 6.900e+01, 6.800e+01, 6.700e+01,

6.600e+01, 6.500e+01, 6.400e+01, 6.300e+01, 6.200e+01, 6.100e+01,

6.000e+01, 5.900e+01, 5.800e+01, 5.700e+01, 5.600e+01, 5.500e+01,

5.400e+01, 5.300e+01, 5.200e+01, 5.100e+01, 5.000e+01, 4.900e+01,

4.800e+01, 4.700e+01, 4.600e+01, 4.500e+01, 4.400e+01, 4.300e+01,

4.200e+01, 4.100e+01, 4.000e+01, 3.900e+01, 3.800e+01, 3.700e+01,

3.600e+01, 3.500e+01, 3.400e+01, 3.300e+01, 3.200e+01, 3.100e+01,

3.000e+01, 2.900e+01, 2.800e+01, 2.700e+01, 2.600e+01, 2.500e+01,

2.400e+01, 2.300e+01, 2.200e+01, 2.100e+01, 2.000e+01, 1.900e+01,

1.800e+01, 1.700e+01, 1.600e+01, 1.500e+01, 1.400e+01, 1.300e+01,

1.200e+01, 1.100e+01, 1.000e+01, 9.000e+00, 8.000e+00, 7.000e+00,

6.000e+00, 5.000e+00, 4.000e+00, 3.000e+00, 2.000e+00, 1.000e+00])

In [50]:



np.unique(cat\_enc\_le)

Out[50]:

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, 59, 60, 61, 62, 63, 64,

65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,

78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90,

91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103,

104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,

117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,

130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,

143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,

156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,

169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,

182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,

195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,

208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,

221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,

234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,

247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259,

260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,

273, 274, 275, 276, 277, 278, 279, 280, 281], dtype=int64)

In [51]:



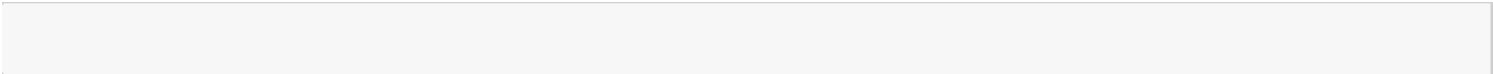
le.inverse\_transform([0, 1, 2, 3])

Out[51]:

array([1., 2., 3., 4.])

**2** 1565.0

In [52]:



ohe = OneHotEncoder()

cat\_enc\_ohe = ohe.fit\_transform(cat\_enc[['c1']])

In [53]:



cat\_enc.shape

Out[53]:

(6896, 1)

In [54]:



cat\_enc\_ohe

Out[54]:

<6896x282 sparse matrix of type '<class 'numpy.float64'>'

with 6896 stored elements in Compressed Sparse Row format>

In [55]:



cat\_enc\_ohe.todense()[0:10]

Out[55]:

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

[0., 0., 0., ..., 0., 1., 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.]])

In [56]:



cat\_enc.head(10)

Out[56]:

**c1**

**0** 3093.0

**1** 2496.0

**2** 1565.0

**3** 1316.0

**4** 1237.0

**5** 1231.0

**6** 1121.0

**7** 1095.0

**8** 1075.0

**9** 1028.0

In [57]:



pd.get\_dummies(cat\_enc).head()

Out[57]:

**c1**

**0** 3093.0

**1** 2496.0



**3** 1316**c**.0**1**

**4** 1237.0

In [58]:



pd.get\_dummies(cat\_temp\_data, dummy\_na=**True**).head()

Out[58]:

**APPEARANCES**

**0** 3093.0

**1** 2496.0

**2** 1565.0

**3** 1316.0

**4** 1237.0

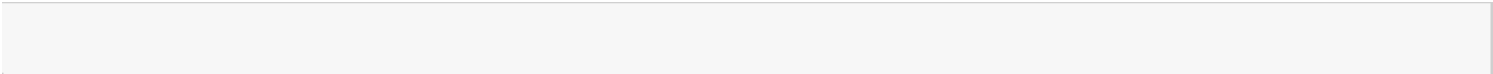
1. **Масштабирование данных**

In [59]:



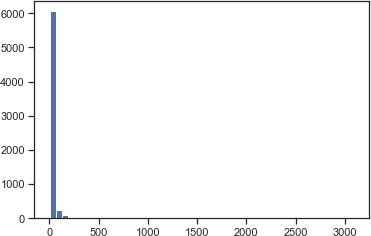
**from sklearn.preprocessing import** MinMaxScaler, StandardScaler, Normalizer

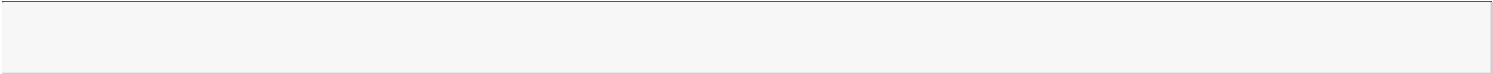
In [61]:



sc1 = MinMaxScaler()

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

In [62]:



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

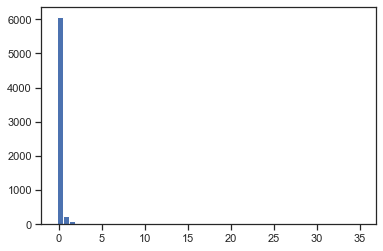
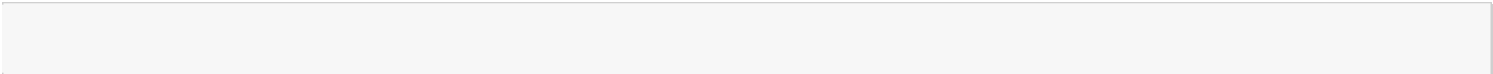
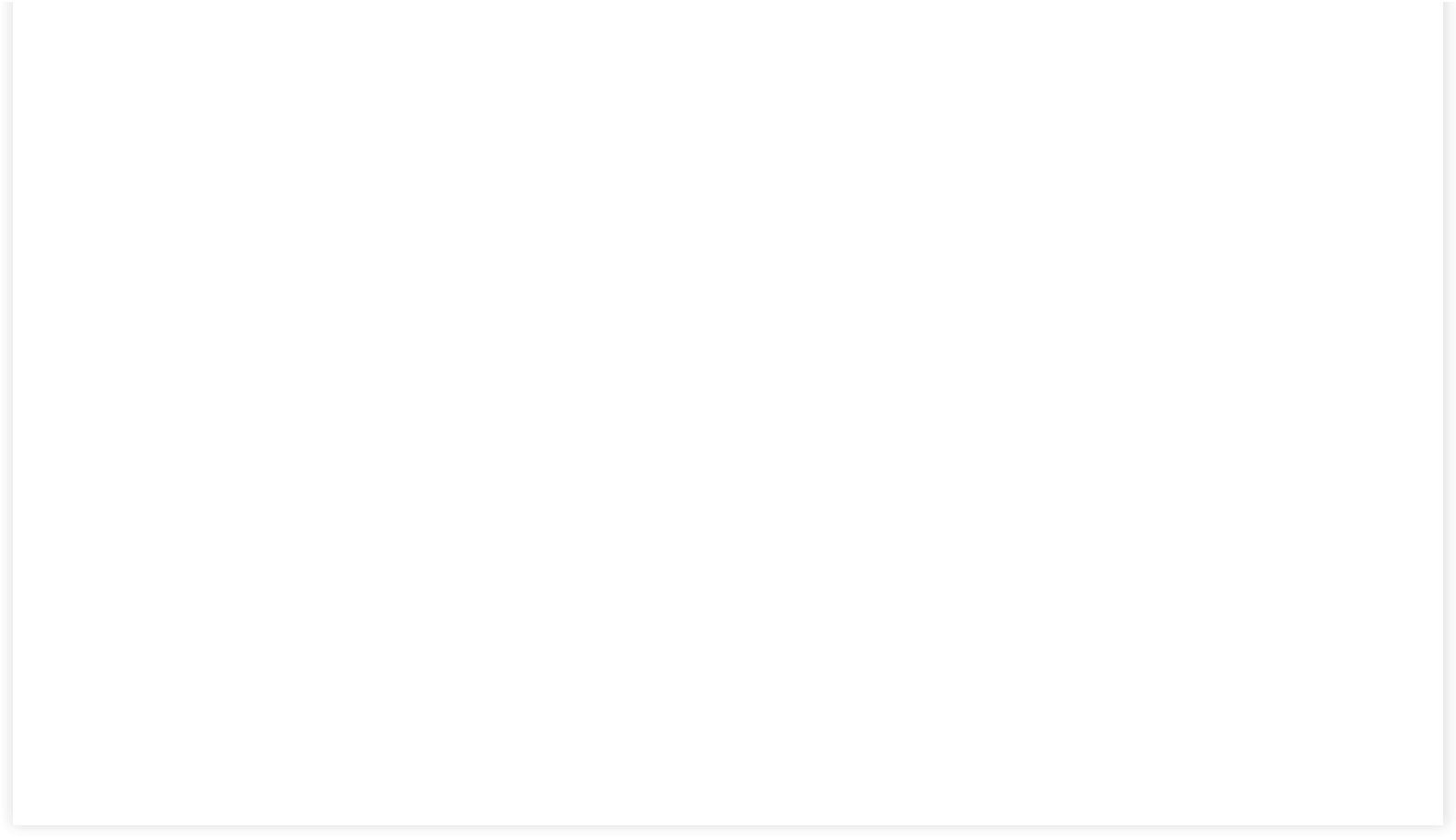
C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\histograms.py:839: RuntimeWarning: invalid value encountered in greater\_equal keep = (tmp\_a >= first\_edge)

C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\histograms.py:840: RuntimeWarning: invalid value encountered in less\_equal keep &= (tmp\_a <= last\_edge)

In [63]:



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



In [65]:

sc2 = StandardScaler()

sc2\_data = sc2.fit\_transform(data[['APPEARANCES']])

In [66]:

plt.hist(sc2\_data, 50) plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\histograms.py:839: RuntimeWarning: invalid value encountered in greater\_equal keep = (tmp\_a >= first\_edge)

C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\histograms.py:840: RuntimeWarning: invalid value encountered in less\_equal keep &= (tmp\_a <= last\_edge)