

Хижняков Вадим, ИУ5-65, Вариант 18

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. Какие методы Вы использовали для решения задачи и почему?

Ввод [1]:

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

Ввод [2]:

```
data = pd.read_csv('./dc-wikia-data.csv', sep=",")
```

Ввод [3]:

```
# размер набора данных
data.shape
```

Out[3]:

```
(6896, 13)
```

Ввод [4]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6896 entries, 0 to 6895
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   page_id               6896 non-null  int64
1   name                  6896 non-null  object
2   urlslug               6896 non-null  object
3   ID                    4883 non-null  object
4   ALIGN                 6295 non-null  object
5   EYE                   3268 non-null  object
6   HAIR                  4622 non-null  object
7   SEX                   6771 non-null  object
8   GSM                   64 non-null    object
9   ALIVE                 6893 non-null  object
10  APPEARANCES           6541 non-null  float64
11  FIRST APPEARANCE      6827 non-null  object
12  YEAR                  6827 non-null  float64
dtypes: float64(2), int64(1), object(10)
memory usage: 700.5+ KB
```

Ввод [5]:

```
# типы колонок
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
```

Ввод [6]:

```
data.head( )
```

Out[6]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	
0	1422	Batman (Bruce Wayne)	VwikiVBatman_(Bruce_Wayne)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Cha
1	23387	Superman (Clark Kent)	VwikiVSuperman_(Clark_Kent)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Cha
2	1458	Green Lantern (Hal Jordan)	VwikiVGreen_Lantern_(Hal_Jordan)	Secret Identity	Good Characters	Brown Eyes	Brown Hair	Cha
3	1659	James Gordon (New Earth)	VwikiVJames_Gordon_(New_Earth)	Public Identity	Good Characters	Brown Eyes	White Hair	Cha
4	1576	Richard Grayson (New Earth)	VwikiVRichard_Grayson_(New_Earth)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Cha

Ввод [7]:

```
# проверим есть ли пропущенные значения
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
```

Ввод [8]:

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

Всего строк: 6896

Ввод [9]:

```
# 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,
```

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%.

Ввод [10]:

```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

Ввод [11]:

```
eye_cat_data = data[['EYE']]
eye_cat_data.head()
```

Out[11]:

	EYE
0	Blue Eyes
1	Blue Eyes
2	Brown Eyes
3	Brown Eyes
4	Blue Eyes

Ввод [12]:

```
eye_cat_data['EYE'].unique()
```

Out[12]:

```
array(['Blue Eyes', 'Brown Eyes', 'Green Eyes', 'Purple Eyes',
      'Black Eyes', 'White Eyes', 'Red Eyes', 'Photocellular Eyes',
      'Hazel Eyes', 'Amber Eyes', 'Yellow Eyes', nan, 'Grey Eyes',
      'Pink Eyes', 'Violet Eyes', 'Gold Eyes', 'Orange Eyes',
      'Auburn Hair'], dtype=object)
```

Ввод [13]:

```
eye_cat_data[eye_cat_data['EYE'].isnull()].shape
```

Out[13]:

```
(3628, 1)
```

Ввод [14]:

```
# Импутация наиболее частыми значениями
imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp = imp.fit_transform(eye_cat_data)
data_imp
```

Out[14]:

```
array([[ 'Blue Eyes'],
      [ 'Blue Eyes'],
      [ 'Brown Eyes'],
      ...,
      [ 'Blue Eyes'],
      [ 'Blue Eyes'],
      [ 'Blue Eyes']], dtype=object)
```

Ввод [15]:

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

Out[15]:

```
array(['Amber Eyes', 'Auburn Hair', 'Black Eyes', 'Blue Eyes',
      'Brown Eyes', 'Gold Eyes', 'Green Eyes', 'Grey Eyes', 'Hazel Eyes',
      'Orange Eyes', 'Photocellular Eyes', 'Pink Eyes', 'Purple Eyes',
      'Red Eyes', 'Violet Eyes', 'White Eyes', 'Yellow Eyes'],
      dtype=object)
```

Ввод [16]:

```
data_imp.shape
```

Out[16]:

```
(6896, 1)
```

Кодирование категорий наборами бинарных значений - one-hot encoding

В этом случае каждое уникальное значение признака становится новым отдельным признаком. Я буду проводить масштабирование при использовании label encoding, но по заданию надо было сделать и one-hot encoding

Ввод [17]:

```
eye_cat_enc = pd.DataFrame({'EYE':data_imp.T[0]})
eye_cat_enc
```

Out[17]:

	EYE
0	Blue Eyes
1	Blue Eyes
2	Brown Eyes
3	Brown Eyes
4	Blue Eyes
...	...
6891	Blue Eyes
6892	Blue Eyes
6893	Blue Eyes
6894	Blue Eyes
6895	Blue Eyes

6896 rows × 1 columns

Ввод [18]:

```
from sklearn.preprocessing import OneHotEncoder
```

Ввод [19] :

```
ohe = OneHotEncoder()  
eye_cat_enc_ohe = ohe.fit_transform(eye_cat_enc[['EYE']])
```

Ввод [20] :

```
eye_cat_enc_ohe.shape
```

Out[20]:

(6896, 17)

Ввод [21]:

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

Out[21]:

[illegible]

Ввод [22]:

```
pd.get_dummies(eye_cat_enc).head()
```

Out[22]:

	EYE_Amber Eyes	EYE_Auburn Hair	EYE_Black Eyes	EYE_Blue Eyes	EYE_Brown Eyes	EYE_Gold Eyes	EYE_Green Eyes	EYE_Grey Eyes
0	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	0
2	0	0	0	0	1	0	0	0
3	0	0	0	0	1	0	0	0
4	0	0	0	1	0	0	0	0

Ввод [23]:

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

Out[23]:

	EYE_Amber Eyes	EYE_Auburn Hair	EYE_Black Eyes	EYE_Blue Eyes	EYE_Brown Eyes	EYE_Gold Eyes	EYE_Green Eyes	EYE_Grey Eyes
0	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	0
2	0	0	0	0	1	0	0	0
3	0	0	0	0	1	0	0	0
4	0	0	0	1	0	0	0	0

Использование LabelEncoder

Был выбран из-за своей ориентации на применение к одному признаку, в нашем случае цвет глаз

Ввод [24]:

```
from sklearn.preprocessing import LabelEncoder
```

Ввод [25]:

```
eye_cat_enc['EYE'].unique()
```

Out[25]:

```
array(['Blue Eyes', 'Brown Eyes', 'Green Eyes', 'Purple Eyes',  
      'Black Eyes', 'White Eyes', 'Red Eyes', 'Photocellular Eyes',  
      'Hazel Eyes', 'Amber Eyes', 'Yellow Eyes', 'Grey Eyes',  
      'Pink Eyes', 'Violet Eyes', 'Gold Eyes', 'Orange Eyes',  
      'Auburn Hair'], dtype=object)
```

Ввод [26]:

```
le = LabelEncoder()  
eye_cat_enc_le = le.fit_transform(eye_cat_enc['EYE'])
```

Ввод [27]:

```
eye_cat_enc_le
```

Out[27]:

```
array([3, 3, 4, ..., 3, 3, 3])
```

Ввод [28]:

```
np.unique(eye_cat_enc_le)
```

Out[28]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,  
16])
```

Ввод [29]:

```
le.inverse_transform([n for n in range(17)])
```

Out[29]:

```
array(['Amber Eyes', 'Auburn Hair', 'Black Eyes', 'Blue Eyes',  
      'Brown Eyes', 'Gold Eyes', 'Green Eyes', 'Grey Eyes', 'Hazel Ey  
es',  
      'Orange Eyes', 'Photocellular Eyes', 'Pink Eyes', 'Purple Eye  
s',  
      'Red Eyes', 'Violet Eyes', 'White Eyes', 'Yellow Eyes'],  
      dtype=object)
```

Ввод [30]:

```
data_digit = eye_cat_enc.copy()  
data_digit["EYE"] = eye_cat_enc_le
```

Масштабирование

Ввод [31]:

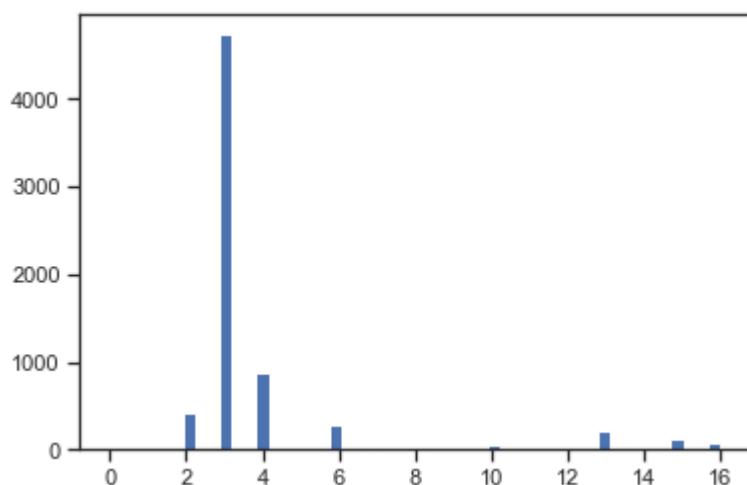
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

Ввод [32]:

```
sc1 = MinMaxScaler()  
sc1_data = sc1.fit_transform(data_digit[['EYE']])
```

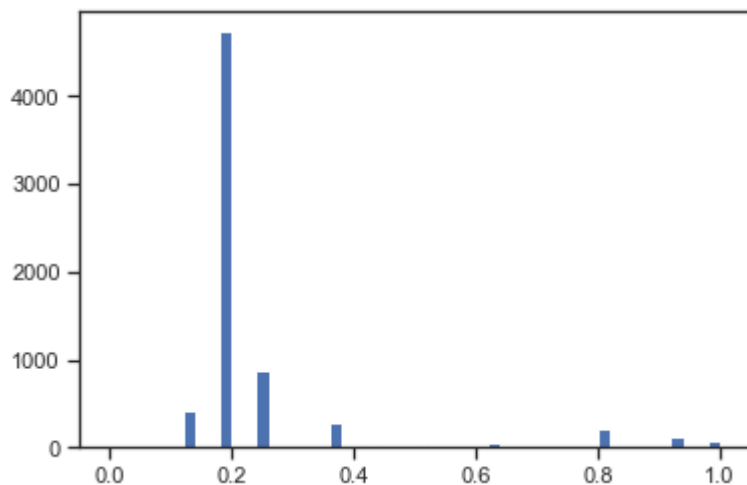

Ввод [33]:

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



Ввод [34]:

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



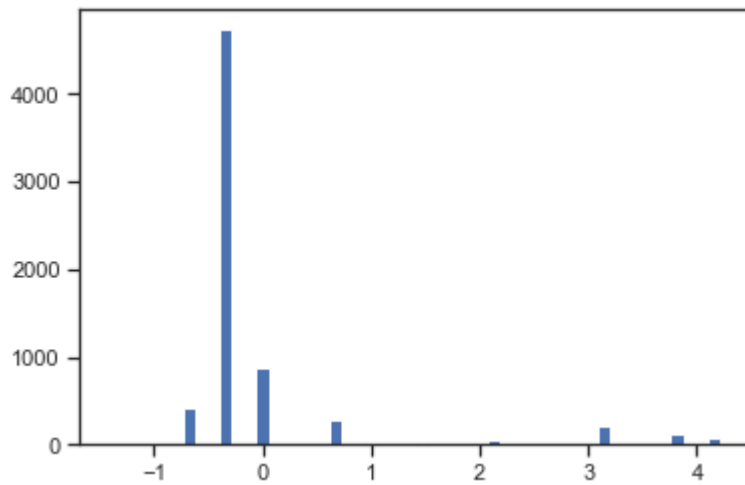
Масштабирование данных на основе Z-оценки - StandardScaler

Ввод [35]:

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

Ввод [36]:

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



Парные диаграммы

Ввод [37]:

```
sns.pairplot(data)
```

Out[37]:

<seaborn.axisgrid.PairGrid at 0x7f863a197e80>

