Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

“Методы машинного обучения”

ЛАБОРАТОРНАЯ РАБОТА № 3. **«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных»**

Студент группы ИУ5-21М

Попков В.Е.

_____ Дата

_____ Подпись

Москва 2019

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.

1) Подключаем библиотеки

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

2) Подключаем БД

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

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	GSM	ALIVE	APPEARANCES	FIRST APPEARANCE	YEAR
0	1422	Batman (Bruce Wayne)	\/wiki\/Batman_(Bruce_Wayne)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	3093.0	1939, May	1939.0
1	23387	Superman (Clark Kent)	\/wiki\/Superman_(Clark_Kent)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	2496.0	1986, October	1986.0
2	1458	Green Lantern (Hal Jordan)	\/wiki\/Green_Lantern_(Hal_Jordan)	Secret Identity	Good Characters	Brown Eyes	Brown Hair	Male Characters	NaN	Living Characters	1565.0	1959, October	1959.0
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, February	1987.0
4	1576	Richard Grayson (New Earth)	\/wiki\/Richard_Grayson_(New_Earth)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	1237.0	1940, April	1940.0

3) Собираем статистику по типу данных и количеству пустых ячеек:

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

```
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
```

```
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
```

4) Обработка пропусков в данных

1.1. Простые стратегии **-** удаление или заполнение нулями

```
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
((6896, 13), (6896, 3))
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
((6896, 13), (38, 13))

data_new_3 = data.fillna(0)
data_new_3.head()
#на данном итервале видна замена Nan значений GSM на 0
```

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	GSM	ALIVE	APPEARANCES	FIRST APPEARANCE	YEAR
0	1422	Batman (Bruce Wayne)	\/wiki\/Batman_(Bruce_Wayne)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	0	Living Characters	3093.0	1939, May	1939.0
1	23387	Superman (Clark Kent)	\/wiki\/Superman_(Clark_Kent)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	0	Living Characters	2496.0	1986, October	1986.0
2	1458	Green Lantern (Hal Jordan)	\/wiki\/Green_Lantern_(Hal_Jordan)	Secret Identity	Good Characters	Brown Eyes	Brown Hair	Male Characters	0	Living Characters	1565.0	1959, October	1959.0
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, February	1987.0
4	1576	Richard Grayson (New Earth)	\/wiki\/Richard_Grayson_(New_Earth)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	0	Living Characters	1237.0	1940, April	1940.0

1.2. "Внедрение значений" **-** импьютация **(imputation)**

1.2.1. Обработка пропусков в числовых данных

```
# Выберем числовые колонки с пропущенными значениями
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)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}.'.format(col, dt,
temp_null_count))
>> Колонка APPEARANCES. Тип данных float64. Количество пустых значений
355.
   Колонка YEAR. Тип данных float64. Количество пустых значений 69.

    # Фильтр по пустым значениям поля MasVnrArea
    data[data['YEAR'].isnull()]
    # Сохраняем индексы
    flt_index = data[data['YEAR'].isnull()].index
    flt_index
>>
Int64Index([386,1400,1401,1832,1937,1938,2065,2066,2067,2230,2231,2232,241
3,2414,2841,2842,3104,3105,3431,3432,3433,3434,3435,3819,3820,3821,3822,38
23,3824,4320,4321,4322,4323,4826,4827,4828,4829,5525,5526,5527,5528,5529,5
530,5531,5532,5533,5534,5535,5536,5537,5538,6532,6533,6534,6535,6536,6537,
6538,6539,6540,6887,6888,6889,6890,6891,6892,6893,6894,6895],dtype=int64)
#заменим значения в этих ячейках медианой по всей выборке
for rows in flt_index:
  data.YEAR[rows]=data.YEAR.median()
#тогда повторный вызов фильтра индексов с пустыми значениями выдаст:
Int64Index([], dtype='int64')
```

1.2.2. Обработка пропусков в категориальных данных

```
# 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 / data.shape[0]) * 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%.
```

Обработаем значения ALIVE, заполнив пропуски наиболее часто встречамым значением:

```
MaxAlive = data.groupby('ALIVE').count()['page_id']
data.ALIVE[data.ALIVE.isnull()]=MaxAlive[MaxAlive== MaxAlive.max()].index[0]
```

1.3 Преобразование категориальных признаков в числовые

#вручную задав параметры

```
data.ALIGN.replace({'Good Characters':'1','Bad Characters':'0'},inplace=True)
data.head()
```

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	GSM	ALIVE	APPEA
0	1422	Batman (Bruce Wayne)	\/wiki\/Batman_(Bruce_Wayne)	Secret Identity	1	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	23.
1	23387	Superman (Clark Kent)	\/wiki\/Superman_(Clark_Kent)	Secret Identity	1	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	23.
2	1458	Green Lantern (Hal Jordan)	\/wiki\/Green_Lantern_(Hal_Jordan)	Secret Identity	1	Brown Eyes	Brown Hair	Male Characters	NaN	Living Characters	23.
3	1659	James Gordon (New Earth)	\/wiki\/James_Gordon_(New_Earth)	Public Identity	1	Brown Eyes	White Hair	Male Characters	NaN	Living Characters	23.
4	1576	Richard Grayson (New Earth)	\/wiki\/Richard_Grayson_(New_Earth)	Secret Identity	1	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	23.

#Средствами LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
dicts = {}

data.ALIGN = label.fit_transform(data.ALIGN.astype(str))
```

```
label.fit(data.ALIGN.drop_duplicates()) #задаем список значений для
кодирования

dicts['ALIGN'] = list(label.classes_)
data.ALIGN = label.transform(data.ALIGN) #заменяем значения из списка кодами
закодированных элементов
flt_index = data['ALIGN'].unique()
flt_index
>>array([1, 0, 2, 4, 3])
```

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	GSM	ALIVE	APPEA
0	1422	Batman (Bruce Wayne)	\/wiki\/Batman_(Bruce_Wayne)	Secret Identity	1	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	23.
1	23387	Superman (Clark Kent)	\/wiki\/Superman_(Clark_Kent)	Secret Identity	1	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	23.
2	1458	Green Lantern (Hal Jordan)	\/wiki\/Green_Lantern_(Hal_Jordan)	Secret Identity	1	Brown Eyes	Brown Hair	Male Characters	NaN	Living Characters	23.
3	1659	James Gordon (New Earth)	\/wiki\/James_Gordon_(New_Earth)	Public Identity	1	Brown Eyes	White Hair	Male Characters	NaN	Living Characters	23.
4	1576	Richard Grayson (New Earth)	\/wiki\/Richard_Grayson_(New_Earth)	Secret Identity	1	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	23.

```
#Средствами pandas - one-hot
import pandas
cat_columns = ['ID']
data_processed = pandas.get_dummies(data, prefix_sep="__",
                                    columns=cat_columns)
data_processed
```

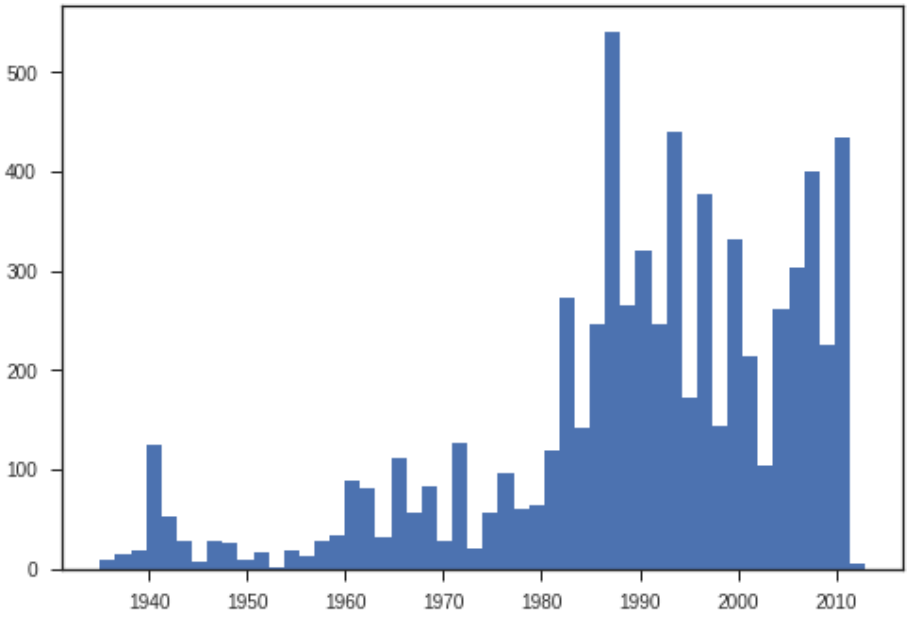
GN	EYE	HAIR	SEX	GSM	ALIVE	APPEARANCES	FIRST APPEARANCE	YEAR	ID__Identity Unknown	ID__Public Identity	ID__Secret Identity
1	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	23.625134	1939, May	1992.0	0	0	1
1	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	23.625134	1986, October	1992.0	0	0	1
1	Brown Eyes	Brown Hair	Male Characters	NaN	Living Characters	23.625134	1959, October	1992.0	0	0	1
1	Brown Eyes	White Hair	Male Characters	NaN	Living Characters	23.625134	1987, February	1992.0	0	1	0
1	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	23.625134	1940, April	1992.0	0	0	1
1	Blue Eyes	Black Hair	Female Characters	NaN	Living Characters	23.625134	1941, December	1992.0	0	1	0
1	Blue Eyes	Blond Hair	Male Characters	NaN	Living Characters	23.625134	1941, November	1992.0	0	1	0

3. Масштабирование данных Термины "масштабирование" и "нормализация" часто используются как синонимы. Масштабирование предполагает изменение диапазона измерения величины, а нормализация - изменение распределения этой величины.

3.1 MinMax масштабирование

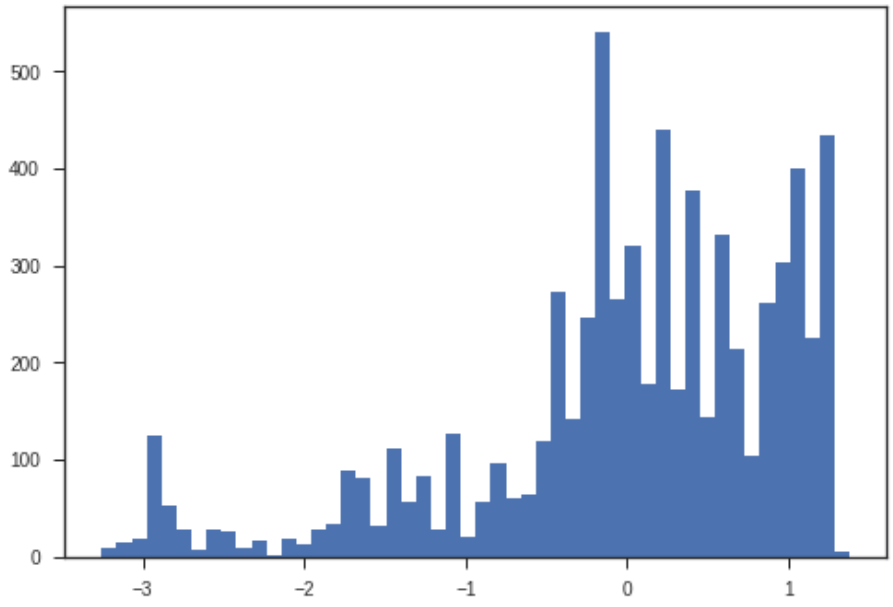
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['YEAR']])
plt.hist(data['YEAR'], 50)
plt.show()
```



3.2. Масштабирование данных на основе **Z**-оценки - **StandardScaler**¶

```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['YEAR']])
plt.hist(sc2_data, 50)
plt.show()
```



3.3 Нормализация данных - **Normalizer**

```
sc3 = Normalizer()
```

```
sc3_data = sc3.fit_transform(data[['YEAR']])
flt_index = data['YEAR'].unique()
flt_index
plt.hist(sc3_data, 50)
plt.show()
```