Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №3
по дисциплине
«Технологии машинного обучения»
на тему
«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных»

Выполнил:
студент группы ИУ5-63Б
Кривцов Н. А.

_____

Москва — 2020 г.

1. Лабораторная работа №3. Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных.

2. Разведочный анализ данных

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in
  ↪the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

[0]:
```
df = pd.read_csv("drive/My Drive/Colab Notebooks/dc-wikia-data.csv")
```

[3]:
```
# Типы признаков
df.dtypes
```

[3]:
```
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]:
```
# Размерность датасета
df.shape
```

[4]:
```
(6896, 13)
```

Все признаки, за исключением APPEARANCES, YEAR и page_id являются категориальаными.

[5]:
```
# Проверка на пропуски в данных
df.isnull().sum()
```

[5]:
```
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
```

3. Обработка пропусков

Для категориального признака GSM значения отсутсвуют почти во всех записях! Признак следует удалить из датасета целиком за его ненадобностью.

[6]:
```
# Удаление признака GSM
df1 = df.drop(labels="GSM", axis=1)
df1.isnull().sum()
```

[6]:
```
page_id                0
name                   0
urlslug                0
ID                  2013
ALIGN                601
EYE                 3628
HAIR                2274
SEX                  125
ALIVE                  3
APPEARANCES          355
FIRST APPEARANCE      69
YEAR                  69
dtype: int64
```

Пропуски в категориальном признаке ALIGN заменим константным значением Unknown.

[0]:
```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=np.nan, strategy='constant',
 ↪fill_value='Unknown')
# Замена исходного столбца на столбец с импутированными пропусками
df1['ALIGN'] = imp.fit_transform(df1[['ALIGN']])
```

Для количественного признака Year (отсутвующих значений - 1%) сгенерируем описательную статистику, а также построим boxplot, чтобы подобрать наиболее подходящую стратегию обработки пропусков.

[8]:
```
df1['YEAR'].describe()
```

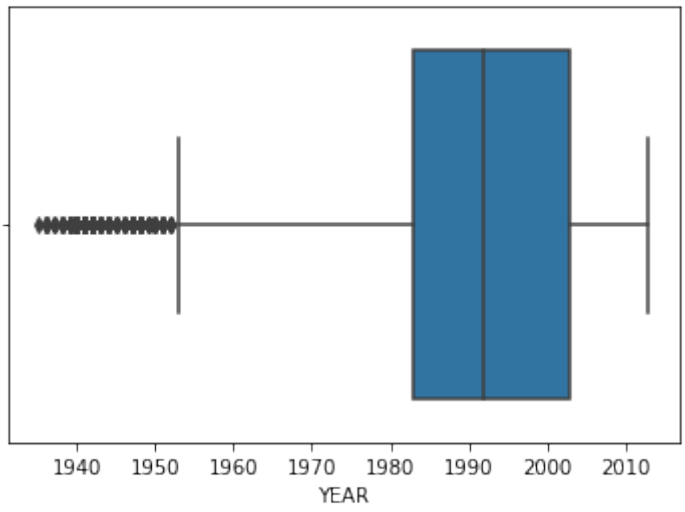
[8]:
```
count    6827.000000
mean     1989.766662
std        16.824194
min      1935.000000
25%      1983.000000
50%      1992.000000
75%      2003.000000
max      2013.000000
Name: YEAR, dtype: float64
```

[9]:
```
df1['YEAR'].value_counts()
```

[9]:
```
2006.0    303
1988.0    286
2010.0    279
1989.0    266
1987.0    254
          ...
1952.0      5
1937.0      4
1935.0      1
1953.0      1
2013.0      1
Name: YEAR, Length: 79, dtype: int64
```

[10]:
```
sns.boxplot(x=df1['YEAR'])
```

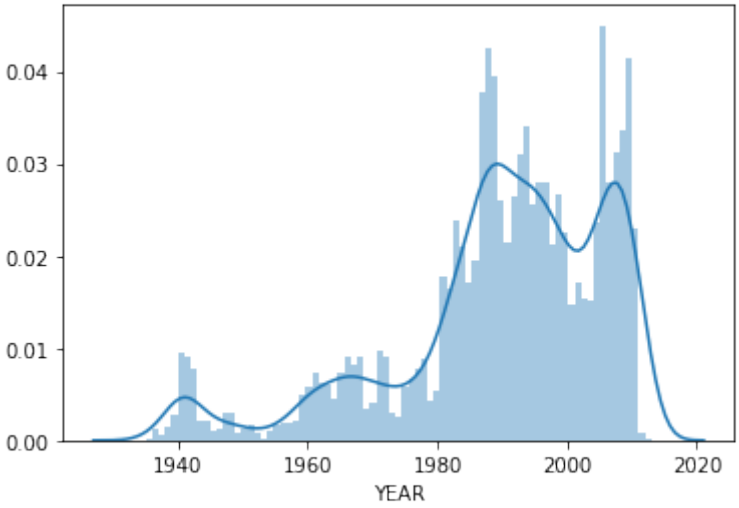
[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93f181fe80>
```



Половина всех значений признака лежит на отрезке [1983, 2003]. Медиана признака YEAR равна 1992.

[11]:
```
sns.distplot(df1['YEAR'], bins=79)
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93ee155470>
```



Заполним пропуски медаинным значением.

[0]:
```
# Медиана
med = df1['YEAR'].median()
```

[13]:
```
df1['YEAR'].tail()
```

[13]:
```
6891   NaN
6892   NaN
6893   NaN
6894   NaN
6895   NaN
Name: YEAR, dtype: float64
```

[14]:
```
# Заполнение пропусков
df1['YEAR'] = df1['YEAR'].fillna(value=med)
df1['YEAR'].tail()
```

[14]:
```
6891    1992.0
6892    1992.0
6893    1992.0
6894    1992.0
6895    1992.0
Name: YEAR, dtype: float64
```

[0]:
```
med = df1['APPEARANCES'].median()
df1['APPEARANCES'] = df1['APPEARANCES'].fillna(value=med)
```

4. Кодирование категориальных признаков

Определим количество всех возможных категорий для каждого признака.

[16]:
```
# Количество категорий по признакам
df1.nunique()
```

[16]:
```
page_id             6896
name                6896
urlslug             6896
ID                     3
ALIGN                  5
EYE                   17
HAIR                  17
SEX                    4
ALIVE                  2
APPEARANCES          282
FIRST APPEARANCE     774
YEAR                  79
dtype: int64
```

Для признака ALIGN определено всего пять категорий. Для его кодирования подойдёт метод **one-shot encoding**.

[17]:
```
# Категории признака ALIGN
df1['ALIGN'].unique()
```

[17]:
```
array(['Good Characters', 'Bad Characters', 'Neutral Characters',
       'Unknown', 'Reformed Criminals'], dtype=object)
```

[18]:
```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder()
enc_cat = df1[['ALIGN']]
enc_cat.shape
```

[18]:
```
(6896, 1)
```

[19]:
```
ohe_enc_cat = ohe.fit_transform(enc_cat)
df3 = pd.DataFrame(ohe_enc_cat.todense(), columns=df1['ALIGN'].unique())
df3
```

[19]:

	Good Characters	Bad Characters	…	Unknown	Reformed Criminals
0	0.0	1.0	…	0.0	0.0
1	0.0	1.0	…	0.0	0.0
2	0.0	1.0	…	0.0	0.0
3	0.0	1.0	…	0.0	0.0
4	0.0	1.0	…	0.0	0.0
…	…	…	…	…	…
6891	0.0	1.0	…	0.0	0.0
6892	0.0	1.0	…	0.0	0.0
6893	0.0	1.0	…	0.0	0.0
6894	0.0	1.0	…	0.0	0.0
6895	1.0	0.0	…	0.0	0.0

[6896 rows x 5 columns]

[20]:
```
# Добавление новых признаков в датасет
df1 = df1.join(df3)
df1
```

[20]:
```
      page_id                          name  … Unknown Reformed
↪Criminals
0        1422           Batman (Bruce Wayne)  …     0.0                0.
↪0
1       23387          Superman (Clark Kent)  …     0.0                0.
↪0
2        1458     Green Lantern (Hal Jordan)  …     0.0                0.
↪0
3        1659       James Gordon (New Earth)  …     0.0                0.
↪0
4        1576    Richard Grayson (New Earth)  …     0.0                0.
↪0
…         …                              …  …     …                  …
6891    66302        Nadine West (New Earth)  …     0.0                0.
↪0
6892   283475     Warren Harding (New Earth)  …     0.0                0.
↪0
6893   283478   William Harrison (New Earth)  …     0.0                0.
↪0
6894   283471   William McKinley (New Earth)  …     0.0                0.
↪0
6895   150660             Mookie (New Earth)  …     0.0                0.
↪0

[6896 rows x 17 columns]
```

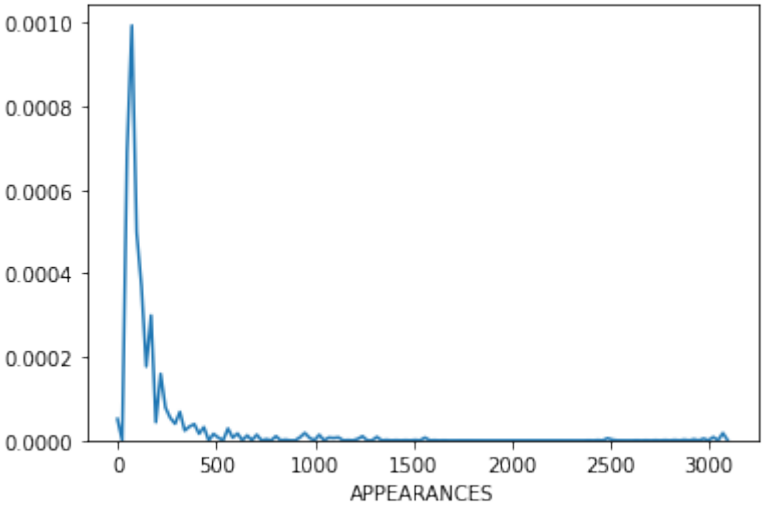
[0]:
```
# Удаление признака ALIGN
del df1['ALIGN']
```

5. Масштабирование данных

Масштабированию подлежит количественный признак APPEARANCES. Используем метод масштабирования по **Z-оценке**.

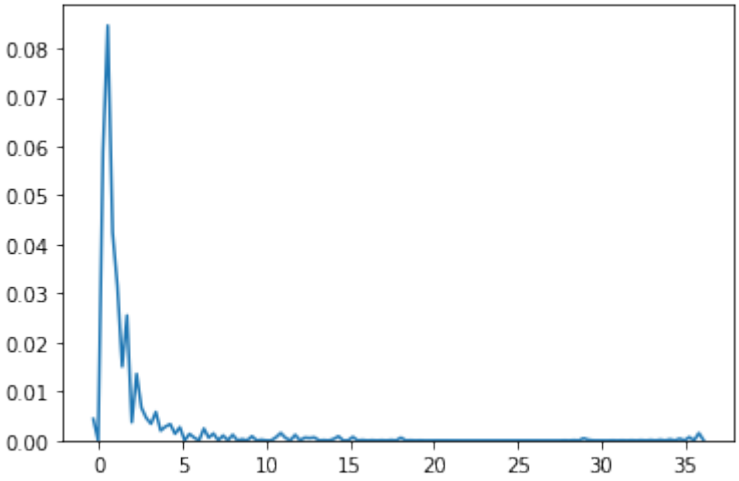
[22]:
```
sns.distplot(df1['APPEARANCES'], hist=False, bins=100)
```

[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93edbee1d0>
```



[23]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df1[['APPEARANCES']])
sns.distplot(scaled_data, hist=False, bins=100)
```

[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f93edafa780>



[24]:
```
df1['APPEARANCES'] = scaled_data
df1
```

[24]:
```
      page_id                          name  … Unknown Reformed
↪Criminals
0        1422          Batman (Bruce Wayne)  …     0.0
↪0
1       23387         Superman (Clark Kent)  …     0.0
↪0
```

```
2        1458    Green Lantern (Hal Jordan)  …     0.0                0.
  ↪0
3        1659      James Gordon (New Earth)  …     0.0                0.
  ↪0
4        1576   Richard Grayson (New Earth)  …     0.0                0.
  ↪0
…         …                              …  …     …                  …
6891    66302       Nadine West (New Earth)  …     0.0                0.
  ↪0
6892   283475    Warren Harding (New Earth)  …     0.0                0.
  ↪0
6893   283478  William Harrison (New Earth)  …     0.0                0.
  ↪0
6894   283471  William McKinley (New Earth)  …     0.0                0.
  ↪0
6895   150660            Mookie (New Earth)  …     0.0                0.
  ↪0

[6896 rows x 16 columns]
```