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 ИУ5-64
Турусов В. И.

Москва — 2019 г.

1. Цель лабораторной работы

Изучить способы предварительной обработки данных для дальнейшего формирования моделей [?].

2. Задание

Требуется [?]:

1. Выбрать набор данных (датасет), содержащий категориальные признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных.
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
   - обработку пропусков в данных;
   - кодирование категориальных признаков;
   - масштабирование данных.

3. Ход выполнения работы

Подключим все необходимые библиотеки и настроим отображение графиков [?, ?]:

[3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import sklearn.impute
import sklearn.preprocessing

# Enable inline plots
%matplotlib inline

# Set plot style
sns.set(style="ticks")

# Set plots formats to save high resolution PNG
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("retina")
```

Зададим ширину текстового представления данных, чтобы в дальнейшем текст в отчёте влезал на A4 [?]:

[ ]:
```
pd.set_option("display.width", 70)
```

Для выполнения данной лабораторной работы возьмём набор данных по персонажам Marvel Comics

[4]:
```
data = pd.read_csv("marvel-data.csv")
```

Посмотрим на эти наборы данных:

[5]:
```
data.head()
```

[5]:
```
   page_id                        name  \
0     1678          Spider-Man (Peter Parker)
1     7139       Captain America (Steven Rogers)
2    64786  Wolverine (James \"Logan\" Howlett)
3     1868    Iron Man (Anthony \"Tony\" Stark)
4     2460                Thor (Thor Odinson)

                            urlslug              ID  \
0           \/Spider-Man_(Peter_Parker)   Secret Identity
1       \/Captain_America_(Steven_Rogers)   Public Identity
2  \/Wolverine_(James_%22Logan%22_Howlett)   Public Identity
3   \/Iron_Man_(Anthony_%22Tony%22_Stark)   Public Identity
4                \/Thor_(Thor_Odinson)  No Dual Identity

              ALIGN        EYE       HAIR            SEX  GSM  \
0    Good Characters  Hazel Eyes  Brown Hair  Male Characters  NaN
1    Good Characters   Blue Eyes  White Hair  Male Characters  NaN
2  Neutral Characters   Blue Eyes  Black Hair  Male Characters  NaN
3    Good Characters   Blue Eyes  Black Hair  Male Characters  NaN
4    Good Characters   Blue Eyes  Blond Hair  Male Characters  NaN

             ALIVE  APPEARANCES FIRST APPEARANCE    Year
0  Living Characters       4043.0         Aug-62  1962.0
1  Living Characters       3360.0         Mar-41  1941.0
2  Living Characters       3061.0         Oct-74  1974.0
3  Living Characters       2961.0         Mar-63  1963.0
4  Living Characters       2258.0         Nov-50  1950.0
```

[6]:
```
data.dtypes
```

[6]:
```
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
```

[7]:
```
data.shape
```

[7]:
```
(16376, 13)
```

3.1. Обработка пропусков в данных

Найдем все пропуски в данных:

[8]:
```
data.isnull().sum()
```

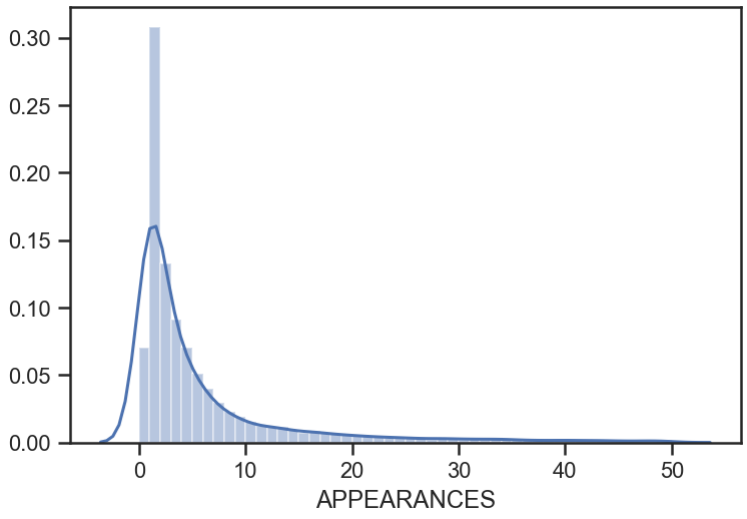
[8]:
```
page_id              0
name                 0
urlslug              0
ID                3770
ALIGN             2812
EYE               9767
HAIR              4264
SEX                854
GSM              16286
ALIVE                3
APPEARANCES       1096
FIRST APPEARANCE   815
Year               815
dtype: int64
```

уберем часть строк, чтобы избавиться от единичных случаев

[40]:
```
data = data.drop(np.where(data['APPEARANCES'] > 50)[0])
```

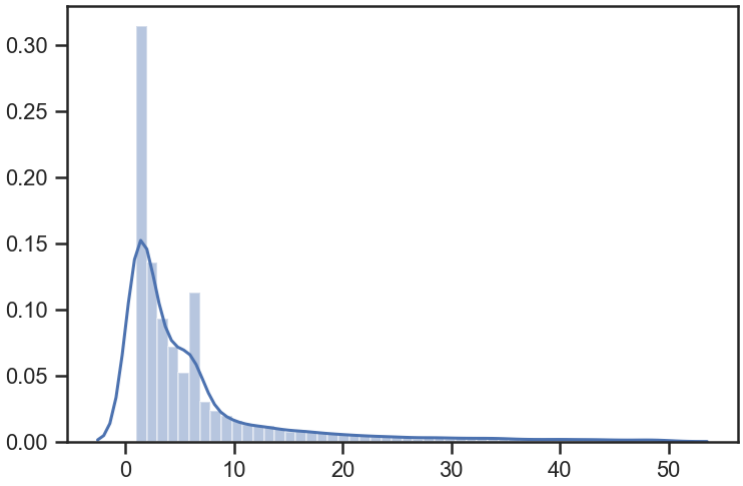
Очевидно, что мы будем работать с колонкой APPEARANCES.
Самый простой вариант — заполнить пропуски нулями:

[21]:
```
sns.distplot(data["APPEARANCES"].fillna(0));
```



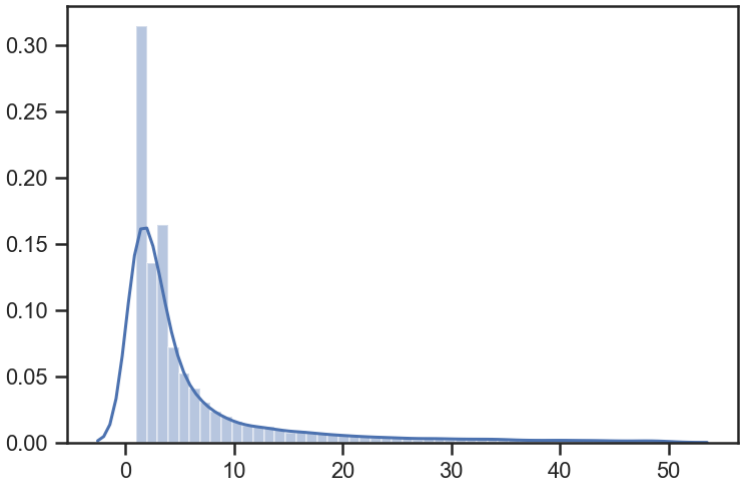
Это работает, но логичнее было бы персонажам без информации присваивать среднее число появлений:

[18]:
```
mean_imp = sklearn.impute.SimpleImputer(strategy="mean")
mean_rat = mean_imp.fit_transform(data[["APPEARANCES"]])
sns.distplot(mean_rat);
```

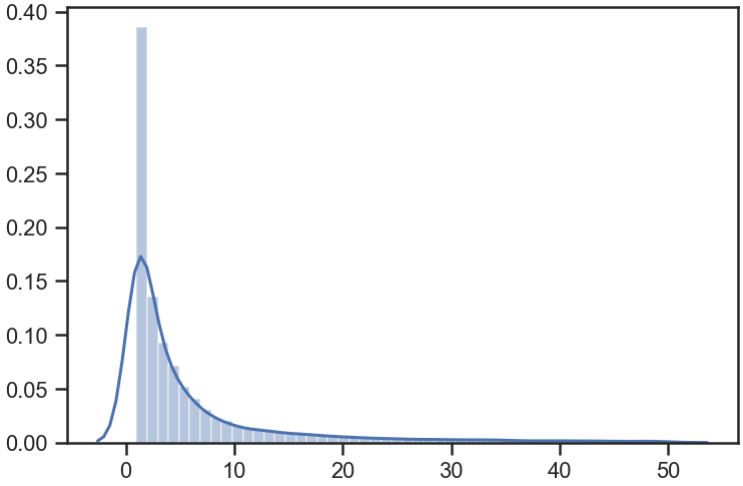


Попробуем также медианное значение и самое частое значение:

[22]:
```
med_imp = sklearn.impute.SimpleImputer(strategy="median")
med_rat = med_imp.fit_transform(data[["APPEARANCES"]])
sns.distplot(med_rat);
```



[23]:
```
freq_imp = sklearn.impute.SimpleImputer(strategy="most_frequent")
freq_rat = freq_imp.fit_transform(data[["APPEARANCES"]])
sns.distplot(freq_rat);
```



Видно, что самый близкий к нормальному распределению график заполнения нулями или самого частого значения. Остановимся на втором:

[25]:
```
data["APPEARANCES"] = freq_rat
```

3.2. Кодирование категориальных признаков

Рассмотрим колонку ID:

[27]:
```
ids = data["ID"].dropna().astype(str)
ids.value_counts()
```

[27]:
```
Secret Identity              5912
Public Identity              4307
No Dual Identity             1649
Known to Authorities Identity    11
Name: ID, dtype: int64
```

Выполним кодирование категорий целочисленными значениями:

[28]:
```
le = sklearn.preprocessing.LabelEncoder()
id_le = le.fit_transform(ids)
print(np.unique(id_le))
le.inverse_transform(np.unique(id_le))
```

```
[0 1 2 3]
```

[28]:
```
array(['Known to Authorities Identity', 'No Dual Identity',
       'Public Identity', 'Secret Identity'], dtype=object)
```

Выполним кодирование категорий наборами бинарных значений:

[29]:
```
id_oh = pd.get_dummies(ids)
id_oh.head()
```

[29]:
```
     Known to Authorities Identity  No Dual Identity  Public Identity  \
735                                0                 0                1
736                                0                 0                0
737                                0                 0                0
738                                0                 0                1
739                                0                 0                1

     Secret Identity
735                0
736                1
737                1
738                0
739                0
```

[30]:
```
id_oh[id_oh["No Dual Identity"] == 1].head()
```

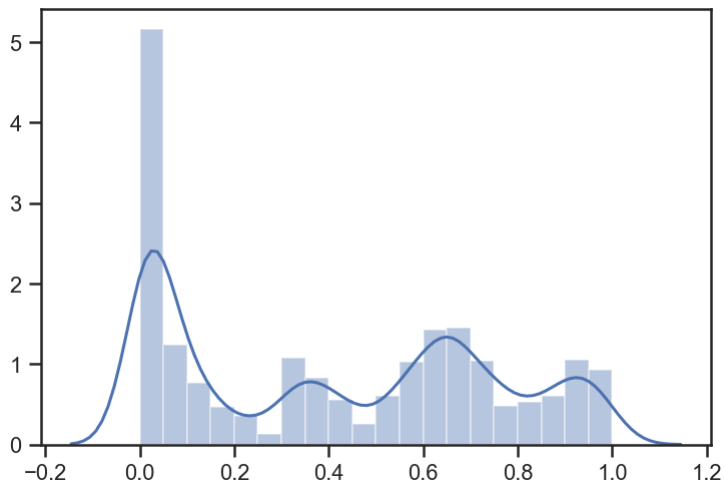
[30]:
```
     Known to Authorities Identity  No Dual Identity  Public Identity  \
740                                0                 1                0
743                                0                 1                0
761                                0                 1                0
763                                0                 1                0
774                                0                 1                0

     Secret Identity
740                0
743                0
761                0
763                0
774                0
```

3.3. Масштабирование данных

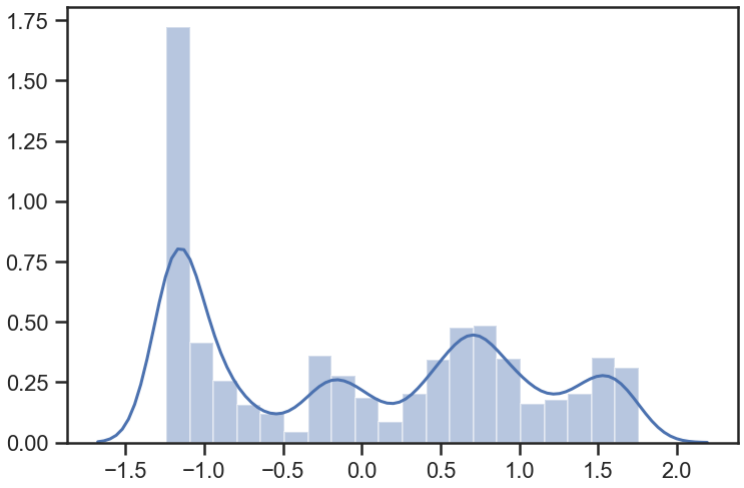
Для начала попробуем обычное MinMax-масштабирование:

[43]:
```
mm = sklearn.preprocessing.MinMaxScaler()
sns.distplot(mm.fit_transform(data[["page_id"]]));
```



Результат вполне ожидаемый и вполне приемлемый. Но попробуем и другие варианты, например, масштабирование на основе Z-оценки:

[50]:
```
ss = sklearn.preprocessing.StandardScaler()
sns.distplot(ss.fit_transform(data[["page_id"]]));
```



Также результат ожидаемый, но его применимость зависит от дальнейшего использования.