

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

Выполнил:
студент группы ИУ5-22М
Смирнов А. И.

1. Цель лабораторной работы

Изучить способы предварительной обработки данных для дальнейшего формирования моделей [1].

2. Задание

Требуется [1]:

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 [2, 3]:

```
[1]: 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 [4]:

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

Для выполнения данной лабораторной работы возьмём набор данных по приложениям в Google Play Store [5]:

```
[3]: data = pd.read_csv("googleplaystore.csv")
```

Посмотрим на эти наборы данных:

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

[4]:

```
App \
0 Photo Editor & Candy Camera & Grid & ScrapBook
1 Coloring book moana
2 U Launcher Lite – FREE Live Cool Themes, Hide ...
3 Sketch - Draw & Paint
4 Pixel Draw - Number Art Coloring Book

Category Rating Reviews Size Installs Type Price \
0 ART_AND_DESIGN 4.1 159 19M 10,000+ Free 0
1 ART_AND_DESIGN 3.9 967 14M 500,000+ Free 0
2 ART_AND_DESIGN 4.7 87510 8.7M 5,000,000+ Free 0
3 ART_AND_DESIGN 4.5 215644 25M 50,000,000+ Free 0
4 ART_AND_DESIGN 4.3 967 2.8M 100,000+ Free 0

Content Rating Genres Last Updated \
0 Everyone Art & Design January 7, 2018
1 Everyone Art & Design;Pretend Play January 15, 2018
2 Everyone Art & Design August 1, 2018
3 Teen Art & Design June 8, 2018
4 Everyone Art & Design;Creativity June 20, 2018

Current Ver Android Ver
0 1.0.0 4.0.3 and up
1 2.0.0 4.0.3 and up
2 1.2.4 4.0.3 and up
3 Varies with device 4.2 and up
4 1.1 4.4 and up
```

[5]: data.dtypes

```
[5]: App      object
Category    object
Rating      float64
Reviews      int64
Size         object
Installs     object
Type         object
Price        object
Content Rating  object
Genres        object
Last Updated   object
Current Ver    object
Android Ver    object
dtype: object
```

[6]: data.shape

[6]: (10841, 13)

3.1. Обработка пропусков в данных

Найдем все пропуски в данных:

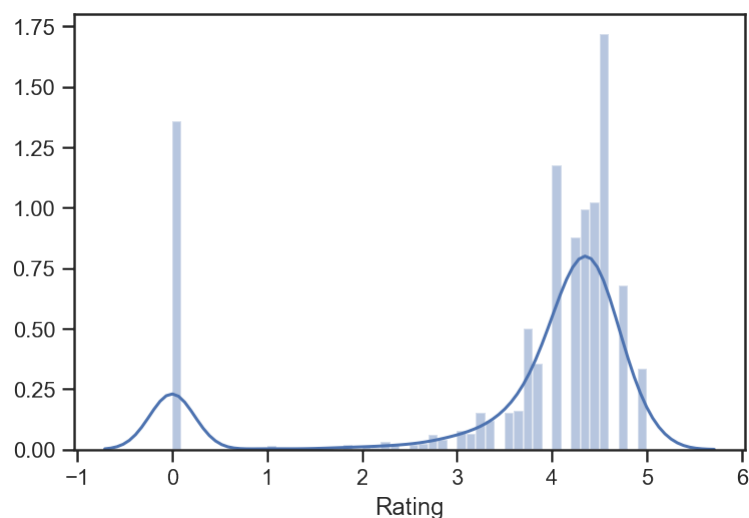
```
[7]: data.isnull().sum()
```

```
[7]: App          0
     Category      0
     Rating      1474
     Reviews       0
     Size          0
     Installs      0
     Type          1
     Price         0
     Content Rating 0
     Genres        1
     Last Updated  0
     Current Ver   8
     Android Ver   2
     dtype: int64
```

Очевидно, что мы будем работать с колонкой **Rating**.

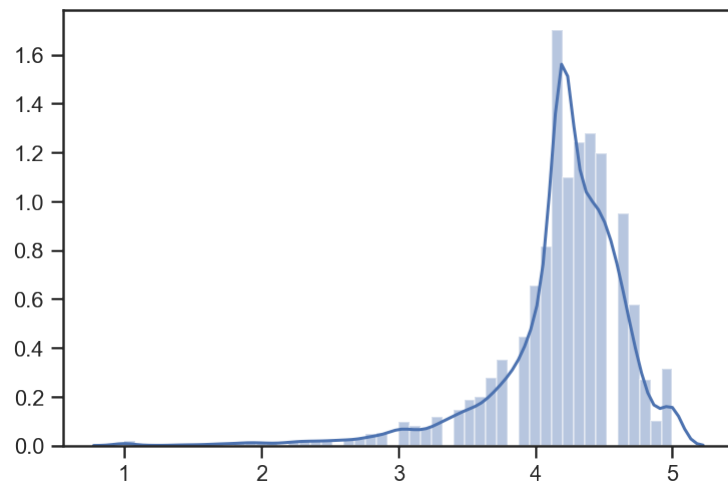
Самый простой вариант — заполнить пропуски нулями:

```
[8]: sns.distplot(data["Rating"].fillna(0));
```



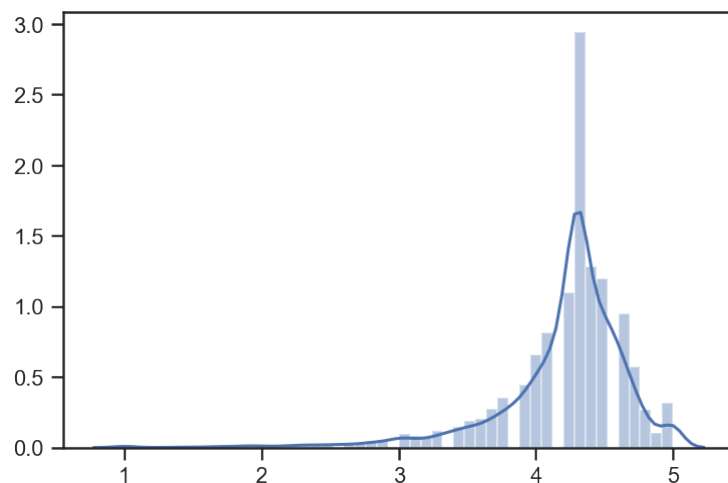
Видно, что в данной ситуации это приводит к выбросам. Логичнее было бы приложениям без рейтинга присваивать средний рейтинг:

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

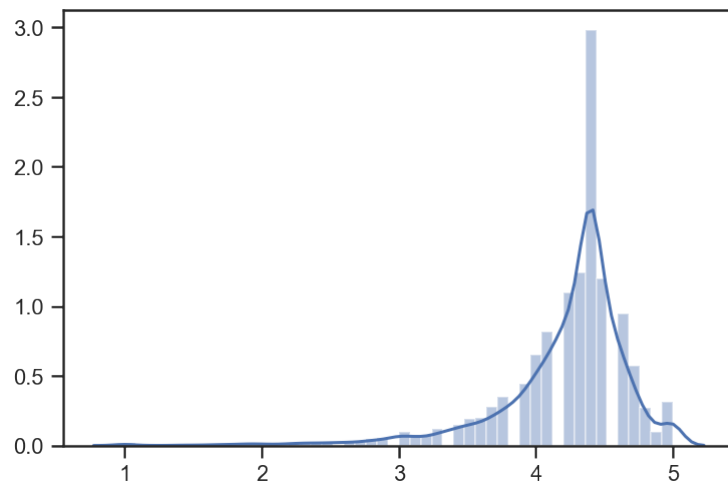


Попробуем также медианный рейтинг и самый частый рейтинг:

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



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



Видно, что самый близкий к нормальному распределению график дало обычное среднее значение. Остановимся на нём:

```
[12]: data["Rating"] = mean_rat
```

3.2. Кодирование категориальных признаков

Рассмотрим колонку Type:

```
[13]: types = data["Type"].dropna().astype(str)
types.value_counts()
```

```
[13]: Free    10040
Paid      800
Name: Type, dtype: int64
```

Выполним кодирование категорий целочисленными значениями:

```
[14]: le = sklearn.preprocessing.LabelEncoder()
type_le = le.fit_transform(types)
print(np.unique(type_le))
le.inverse_transform(np.unique(type_le))
```

```
[0 1]
```

```
[14]: array(['Free', 'Paid'], dtype=object)
```

Выполним кодирование категорий наборами бинарных значений:

```
[15]: type_oh = pd.get_dummies(types)
type_oh.head()
```

```
[15]:   Free Paid
0     1    0
1     1    0
2     1    0
3     1    0
```

4 1 0

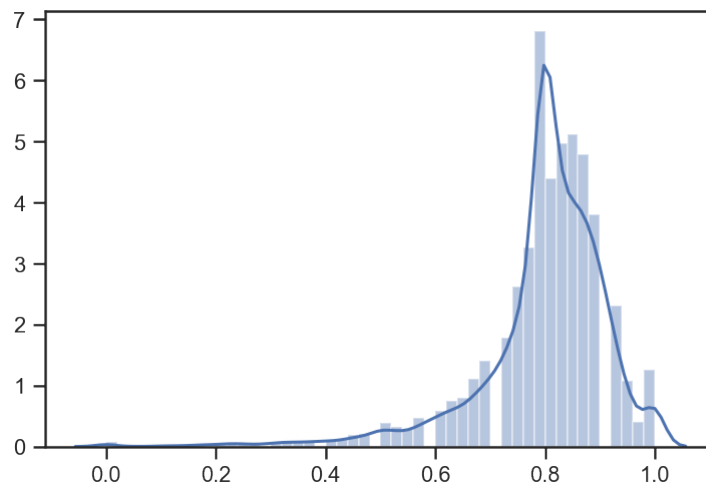
```
[16]: type_oh[type_oh["Paid"] == 1].head()
```

```
[16]:   Free Paid
      234  0  1
      235  0  1
      290  0  1
      291  0  1
      427  0  1
```

3.3. Масштабирование данных

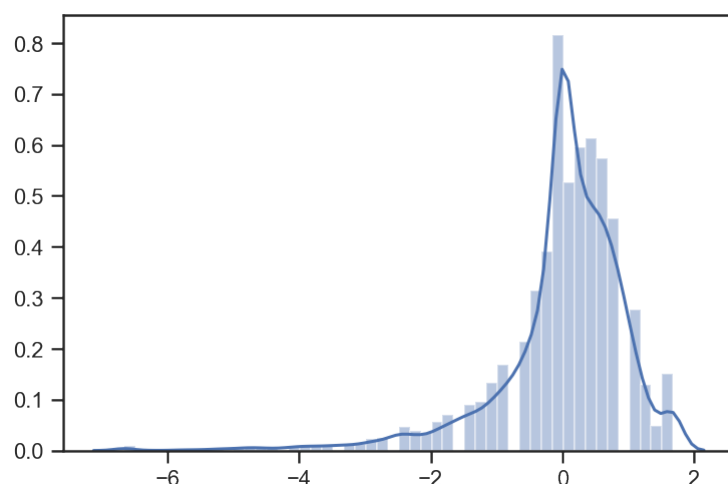
Для начала попробуем обычное MinMax-масштабирование:

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



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

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



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

Также была опробована нормализация данных, но единственным результатом была ошибка `LinAlgError: singular matrix`. С чем она связана не до конца очевидно, вероятно, метод `sklearn.preprocessing.Normalizer` плохо рассчитан на одноколоночные данные.

Список литературы

- [1] Гапанюк Ю. Е. Лабораторная работа «Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_MISSING (дата обращения: 05.04.2019).
- [2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. — 2019. — Access mode: <https://ipython.readthedocs.io/en/stable/> (online; accessed: 20.02.2019).
- [3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. — Access mode: <https://seaborn.pydata.org/> (online; accessed: 20.02.2019).
- [4] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: <http://pandas.pydata.org/pandas-docs/stable/> (online; accessed: 20.02.2019).
- [5] Gupta L. Google Play Store Apps [Electronic resource] // Kaggle. — 2019. — Access mode: <https://www.kaggle.com/lava18/google-play-store-apps> (online; accessed: 05.04.2019).