Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Автоматизированные системы обработки информации и управления»



**Отчет**

**по лабораторной работе № 3**

**по курсу «Технологии машинного обучения»**

**«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование**

**данных.»**

**ИСПОЛНИТЕЛЬ:**

Мехмандаров Мурад

Группа ИУ5-63

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ "\_\_"\_\_\_\_\_\_\_\_\_\_\_2020 г.

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ "\_\_"\_\_\_\_\_\_\_\_\_\_\_2020 г.

Москва 2020

# 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.

# Дата-сет

In

[17]:

Out[17]:

**import**

numpy

**as**

np

**import**

pandas

**as**

pd

**import**

seaborn

**as**

sns

**import**

sklearn

.

impute

**from**

sklearn

.

preprocessing

**import**

LabelEncoder

,

MinMaxScaler

,

StandardScaler

data

**=**

pd

.

read\_csv

(

"../data/games.csv"

)

data

.

head

()

**Name Platform Year\_of\_Release Genre Publisher NA\_Sales EU\_Sales JP\_Sales**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Wii Sports | Wii | 2006.0 | Sports | Nintendo | 41.36 | 28.96 | 3.77 |
| **1** | Super Mario Bros. | NES | 1985.0 | Platform | Nintendo | 29.08 | 3.58 | 6.81 |
| **2** | Mario Kart Wii | Wii | 2008.0 | Racing | Nintendo | 15.68 | 12.76 | 3.79 |
| **3** | Wii Sports Resort | Wii | 2009.0 | Sports | Nintendo | 15.61 | 10.93 | 3.28 |
| **4** | Pokemon Red/Pokemon | GB | 1996.0 | RolePlaying | Nintendo | 11.27 | 8.89 | 10.22 |

Blue

Посмотрим на типы колонок

[4]: data.dtypes

Out[4]:

Name object

Platform object

Year\_of\_Release float64

Genre object

Publisher object

NA\_Sales float64

EU\_Sales float64

JP\_Sales float64

Other\_Sales float64

Global\_Sales float64

Critic\_Score float64

Critic\_Count float64

User\_Score object

User\_Count float64

Developer object Rating object dtype: object In [16]:

data

.

shape

Out[16]:

(16719, 16)

**## Обработка пропусков в данных**

In [17]:

data

.

isnull

().

sum

()

Out[17]:

Name 2

Platform 0

Year\_of\_Release 269

Genre 2

Publisher 54

NA\_Sales 0

EU\_Sales 0

JP\_Sales 0

Other\_Sales 0

Global\_Sales 0

Critic\_Score 8582

Critic\_Count 8582

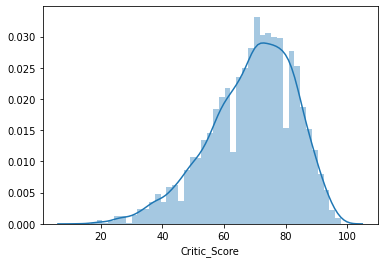
User\_Score 6704

User\_Count 9129

Developer 6623 Rating 6769 dtype: int64

[21]:

sns.distplot(data["Critic\_Score"]);



In [52]:

sns

.

distplot

(

data

[

"Critic\_Score"

].

fillna

(

0

))

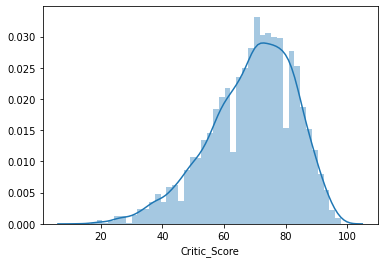
;

Out[52]:

1. 76.0
2. 0.0
3. 82.0
4. 80.0
5. 0.0 ...
6. 0.0
7. 0.0
8. 0.0
9. 0.0
10. 0.0

Name: Critic\_Score, Length: 16719, dtype: float64 [59]:

sns.distplot(data["Critic\_Score"].dropna());

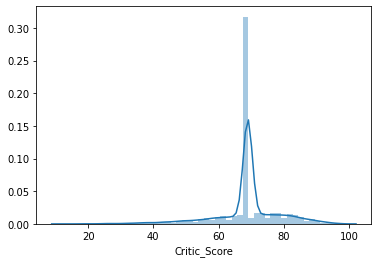


In [56]:

sns.distplot(data["Critic\_Score"].fillna(data["Critic\_Score"].mean()))

Out[56]:

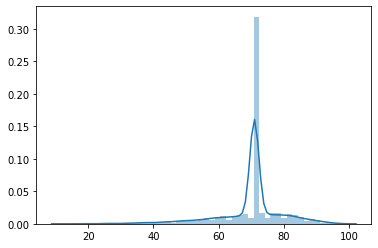
<matplotlib.axes.\_subplots.AxesSubplot at 0x11fbb8c90>



[12]:

In

[5]:



mediana

**=**

sklearn

.

impute

.

SimpleImputer

(

strategy

**=**

"median"

)

median\_rating

**=**

mediana

.

fit\_transform

(

data

[[

"Critic\_Score"

]])

sns

.

distplot

(

median\_rating

)

;

sns

.

distplot

(

data

[

"Critic\_Score"

].

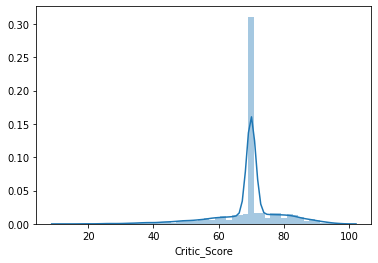
fillna

(

70.0

))

Out[5]: <matplotlib.axes.\_subplots.AxesSubplot at 0x121cd9e50>



In [13]:

data

[

"Critic\_Score"

]

**=**

median\_rating

data

[

"Critic\_Score"

].

isnull

().

sum

()

Out[13]:

0

Как видим, у колонки Rating больше нет пропущенных значений

# Кодирование категориальных признаков

[15]:

categories

**=**

data

[

"Genre"

].

dropna

().

astype

(

str

)

categories

.

value\_counts

(

)

Out[15]:

Action 3370

Sports 2348

Misc 1750

Role-Playing 1500

Shooter 1323

Adventure 1303

Racing 1249

Platform 888

Simulation 874

Fighting 849

Strategy 683

Puzzle 580 Name: Genre, dtype: int64

In

[18]:

0 1 2 3 4 5 6 7 8 9

[

10 11]

le

**=**

LabelEncoder

()

category\_le

**=**

le

.

fit\_transform

(

categories

)

print

(

np

.

unique

(

category\_le

))

le

.

inverse\_transform

(

np

.

unique

(

category\_le

))

Out[18]:

array(['Action', 'Adventure', 'Fighting', 'Misc', 'Platform', 'Puzzl e',

'Racing', 'Role-Playing', 'Shooter', 'Simulation', 'Sports', 'Strategy'], dtype=object) In [19]:

data

.

head

(

)

Out[19]:

**Name Platform Year\_of\_Release Genre Publisher NA\_Sales EU\_Sales JP\_Sales**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Wii Sports | Wii | 2006.0 | Sports | Nintendo | 41.36 | 28.96 | 3.77 |
| **1** | Super Mario Bros. | NES | 1985.0 | Platform | Nintendo | 29.08 | 3.58 | 6.81 |
| **2** | Mario Kart Wii | Wii | 2008.0 | Racing | Nintendo | 15.68 | 12.76 | 3.79 |
| **3** | Wii Sports Resort | Wii | 2009.0 | Sports | Nintendo | 15.61 | 10.93 | 3.28 |
| **4** | Pokemon Red/Pokemon | GB | 1996.0 | RolePlaying | Nintendo | 11.27 | 8.89 | 10.22 |

Blue

# Масштабирование данных

min-max

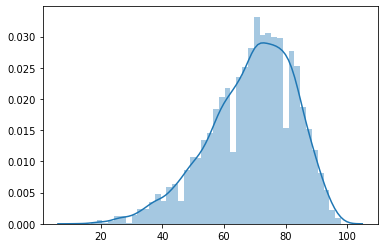
масштабирование

In

[20]:

In

[21]:



sns

.

distplot

(

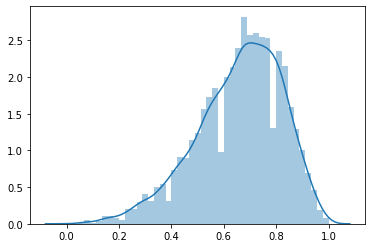
data

[[

"Critic\_Score"

]])

;



mm

**=**

MinMaxScaler

()

sns

.

distplot

(

mm

.

fit\_transform

(

data

[[

"Critic\_Score"

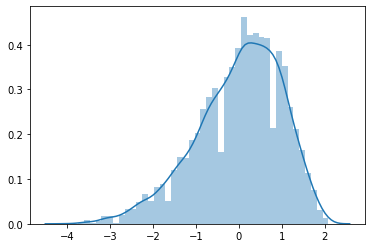
]]))

;

На основе Z-оценки

In

[22]:



ss

**=**

StandardScaler

()

sns

.

distplot

(

ss

.

fit\_transform

(

data

[[

"Critic\_Score"

]]))

;