

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ  
им. Н.Э. Баумана

Факультет «Информатика и системы управления»  
Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Лабораторная работа №\_\_2\_\_  
по дисциплине «Методы машинного обучения»

Тема: «Обработка признаков (часть 1).»

ИСПОЛНИТЕЛЬ:

группа

Коротков Н.К.

ФИО

ИУ5-23М

подпись

"\_\_"\_\_2024 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

ФИО

подпись

"\_\_"\_\_2024 г.

Москва - 2024

---

## **Задание**

1. Выбрать набор данных (датасет), содержащий категориальные и числовые 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.) Просьба не использовать датасет, на котором данная задача решалась в лекции.

2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

- a. устранение пропусков в данных;
- b. кодирование категориальных признаков;
- c. нормализация числовых признаков.

3. Сформировать отчет и разместить его в своем репозитории на github.

# Лабораторная работа 2

датасет 1 <https://www.kaggle.com/datasets/lava18/google-play-store-apps>

датасет 2 <https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who>

датасет 3 <https://www.kaggle.com/datasets/muthuj7/weather-dataset>

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
import scipy.stats as stats

data = pd.read_csv(r'C:\Users\ksarb\Documents\MMO_2024\Datasets\
googleplaystore.csv', sep=",")

data.isnull().sum()

App                0
Category           0
Rating            1474
Reviews           0
Size              0
Installs          0
Type              1
Price             0
Content Rating     1
Genres            0
Last Updated      0
Current Ver       8
Android Ver       3
dtype: int64

data.shape

(10841, 13)

data.head()
```

	App	Category
Rating \		
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART AND DESIGN

```

4.1
1          Coloring book moana  ART_AND_DESIGN
3.9
2  U Launcher Lite - FREE Live Cool Themes, Hide ...  ART_AND_DESIGN
4.7
3          Sketch - Draw & Paint  ART_AND_DESIGN
4.5
4          Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN
4.3

```

	Reviews	Size	Installs	Type	Price	Content Rating	\
0	159	19M	10,000+	Free	0	Everyone	
1	967	14M	500,000+	Free	0	Everyone	
2	87510	8.7M	5,000,000+	Free	0	Everyone	
3	215644	25M	50,000,000+	Free	0	Teen	
4	967	2.8M	100,000+	Free	0	Everyone	

	Genres	Last Updated	Current Ver	\
0	Art & Design	January 7, 2018	1.0.0	
1	Art & Design;Pretend Play	January 15, 2018	2.0.0	
2	Art & Design	August 1, 2018	1.2.4	
3	Art & Design	June 8, 2018	Varies with device	
4	Art & Design;Creativity	June 20, 2018	1.1	

	Android Ver
0	4.0.3 and up
1	4.0.3 and up
2	4.0.3 and up
3	4.2 and up
4	4.4 and up

Пропуски в данных в столбцах type, content\_rating и ver (оба) можно обработать удалением - это единичные значения.

```

colsForDel = ['Type', 'Content Rating', 'Current Ver', 'Android Ver']
data_drop_na = data[colsForDel].dropna()
data_drop_na.shape

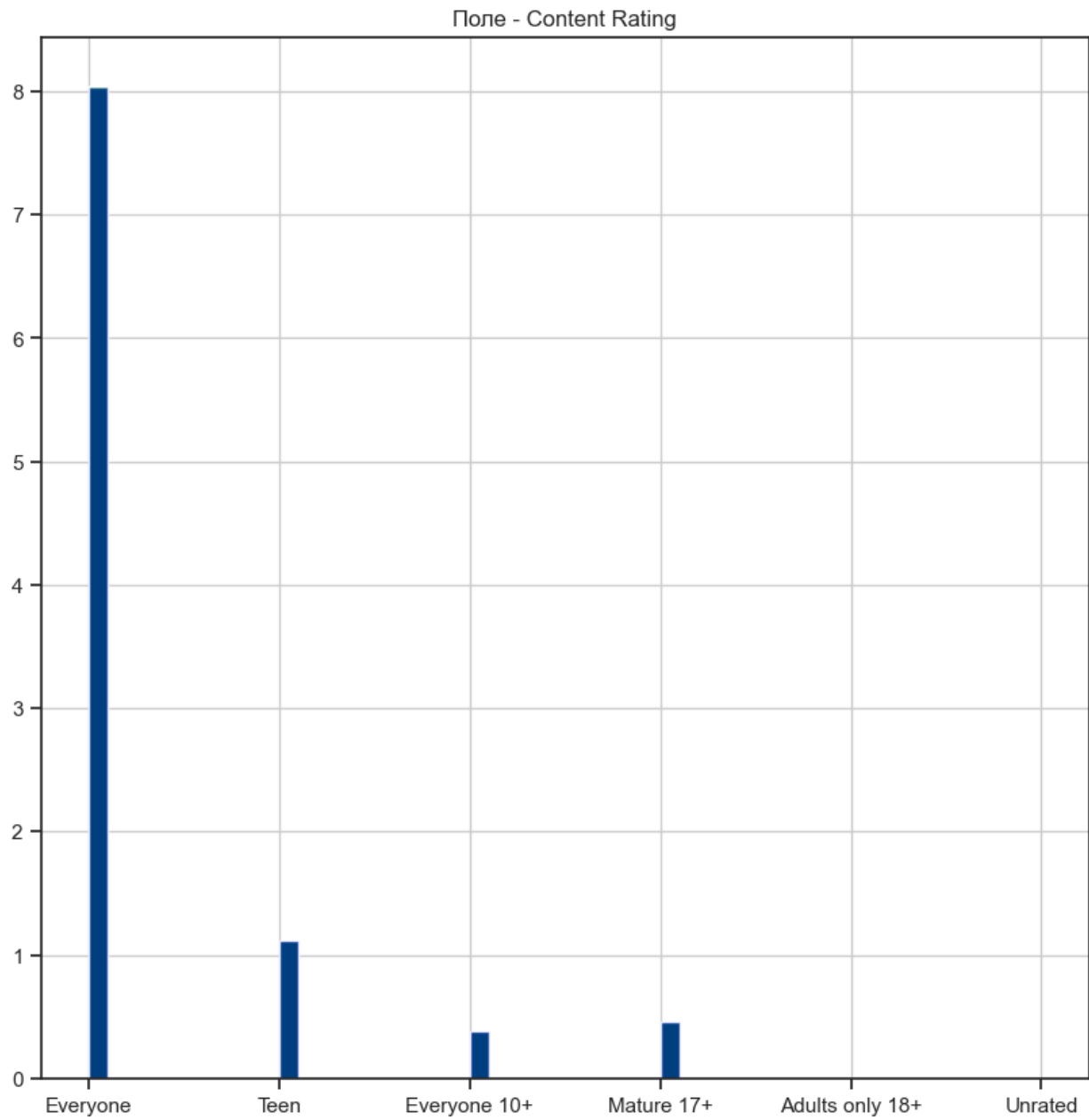
(10829, 4)

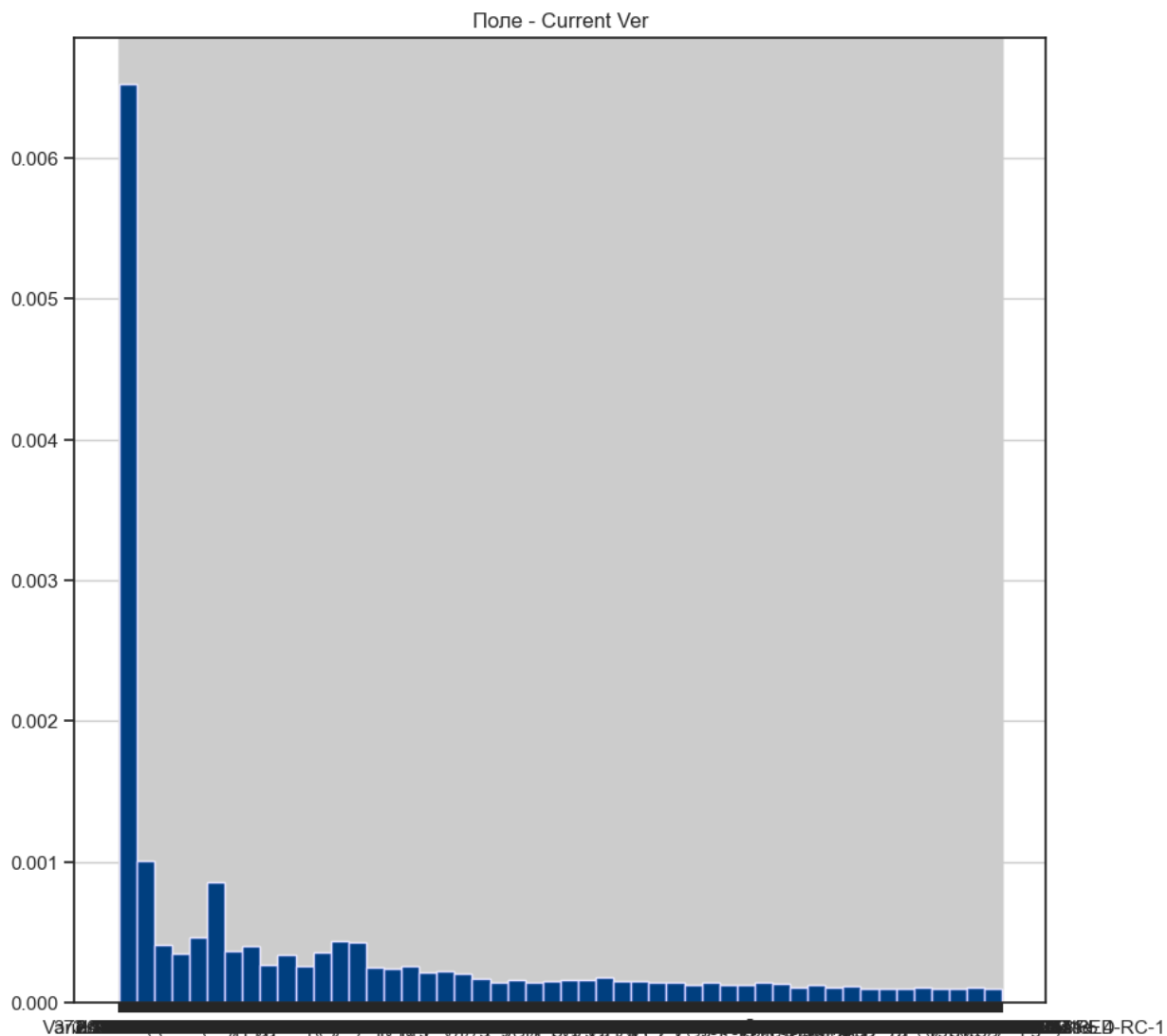
def plot_hist_diff(old_ds, new_ds, cols):
    """
    Разница между распределениями до и после устранения пропусков
    """
    for c in cols:
        fig, ax = plt.subplots(figsize=(10,10))
        ax.title.set_text('Поле - ' + str(c))
        old_ds[c].hist(bins=50, ax=ax, density=True, color='green')
        new_ds[c].hist(bins=50, ax=ax, color='blue', density=True,

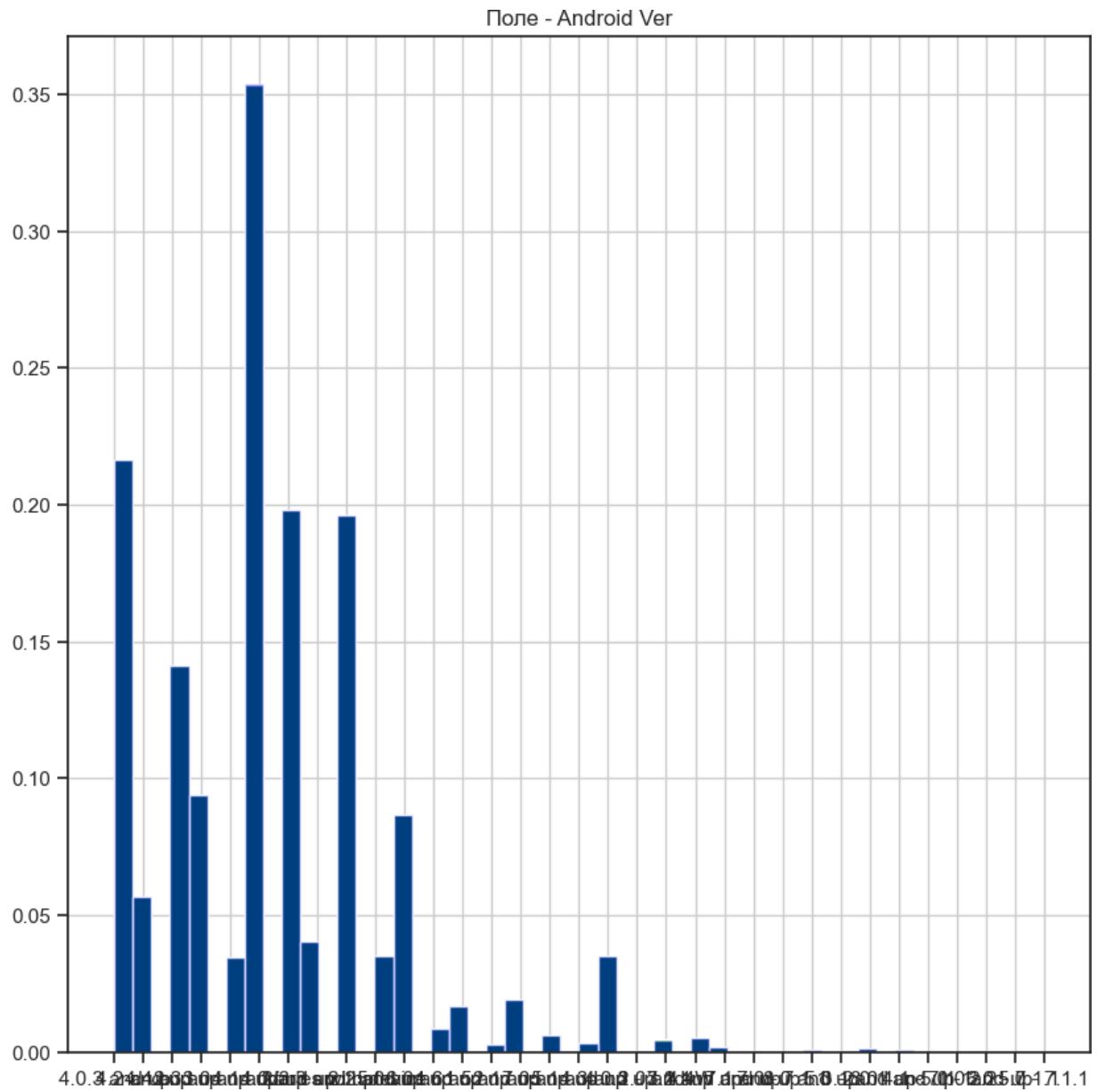
```

```
alpha=0.5)
    plt.show()

plot_hist_diff(data, data_drop_na, ['Content Rating', 'Current Ver',
'Android Ver'])
```







```
data.dtypes
```

App	object
Category	object
Rating	float64
Reviews	object
Size	object
Installs	object
Type	object
Price	object
Content Rating	object
Genres	object

```
Last Updated      object
Current Ver       object
Android Ver       object
dtype: object
```

```
data = data.dropna(subset=colsForDel)
data.shape
```

```
(10829, 13)
```

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

```
App                0
Category           0
Rating            1469
Reviews            0
Size               0
Installs           0
Type               0
Price              0
Content Rating     0
Genres             0
Last Updated       0
Current Ver        0
Android Ver        0
dtype: int64
```

```
fig, ax = plt.subplots(figsize=(5,5))
sns.distplot(data['Rating'])
```

```
C:\Users\ksarb\AppData\Local\Temp\ipykernel_3944\195523562.py:2:
UserWarning:
```

```
`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.
```

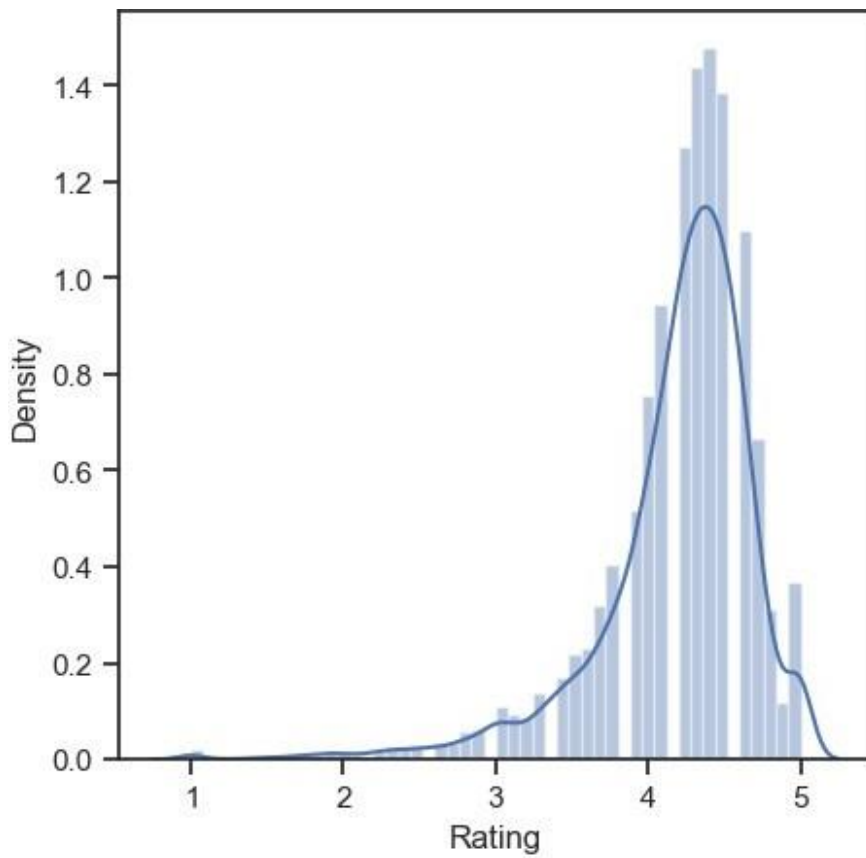
Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(data['Rating'])
```

```
<Axes: xlabel='Rating', ylabel='Density'>
```





Заполним rating

```
def impute_column(dataset, column, strategy_param,
fill_value_param=None):
    """
    Заполнение пропусков в одном признаке
    """
    temp_data = dataset[[column]].values
    size = temp_data.shape[0]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imputer = SimpleImputer(strategy=strategy_param,
                             fill_value=fill_value_param)
    all_data = imputer.fit_transform(temp_data)

    missed_data = temp_data[mask_missing_values_only]
    filled_data = all_data[mask_missing_values_only]

    return all_data.reshape((size,)), filled_data, missed_data

filled_data, _, _ = impute_column(data, 'Rating', 'median')
```

```
fig, ax = plt.subplots(figsize=(5,5))
sns.distplot(filled_data)
```

C:\Users\ksarb\AppData\Local\Temp\ipykernel\_3944\2113822577.py:2:  
UserWarning:

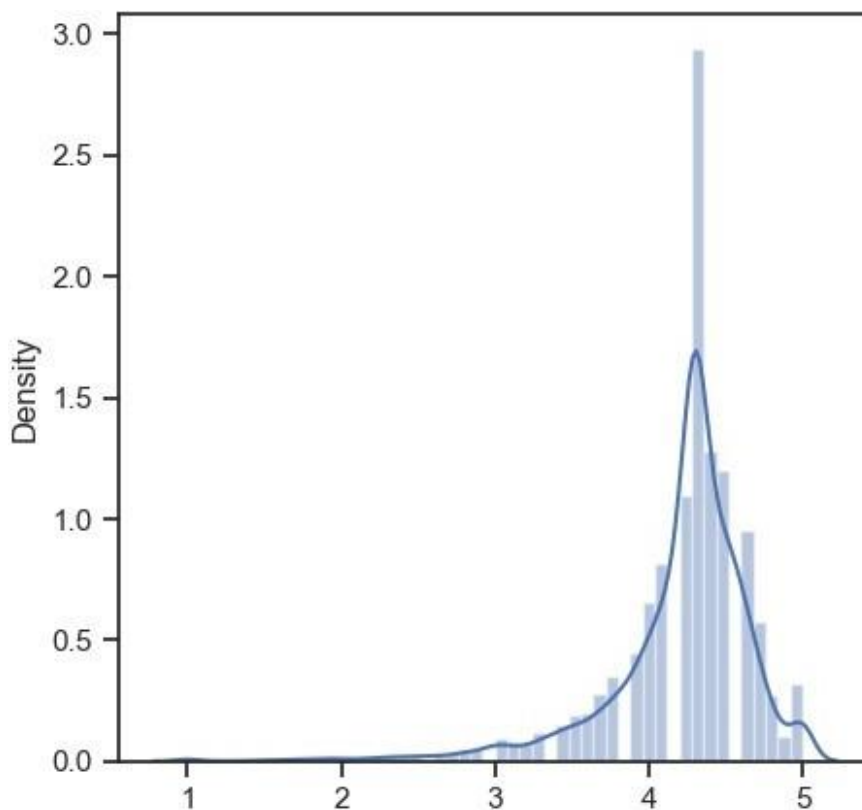
`distplot` is a deprecated function and will be removed in seaborn  
v0.14.0.

Please adapt your code to use either `displot` (a figure-level  
function with  
similar flexibility) or `histplot` (an axes-level function for  
histograms).

For a guide to updating your code to use the new functions, please see  
<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(filled_data)
```

<Axes: ylabel='Density'>



```
filled_data
```

```
array([4.1, 3.9, 4.7, ..., 4.3, 4.5, 4.5])
```

```

knnimpute_hdata = data[['Reviews', 'Rating']].copy()
knnimpute_hdata.head()
from sklearn.impute import KNNImputer
knnimputer = KNNImputer(
    n_neighbors=5,
    weights='distance',
    metric='nan_euclidean',
    add_indicator=False,
)
knnimpute_hdata_imputed_temp =
knnimputer.fit_transform(knnimpute_hdata)
knnimpute_hdata_imputed = pd.DataFrame(knnimpute_hdata_imputed_temp,
columns=knnimpute_hdata.columns)
knnimpute_hdata_imputed.head()

```

	Reviews	Rating
0	159.0	4.1
1	967.0	3.9
2	87510.0	4.7
3	215644.0	4.5
4	967.0	4.3

```

fig, ax = plt.subplots(figsize=(5,5))
sns.distplot(knnimpute_hdata['Rating'])

```

C:\Users\ksarb\AppData\Local\Temp\ipykernel\_3944\274606484.py:2:  
UserWarning:

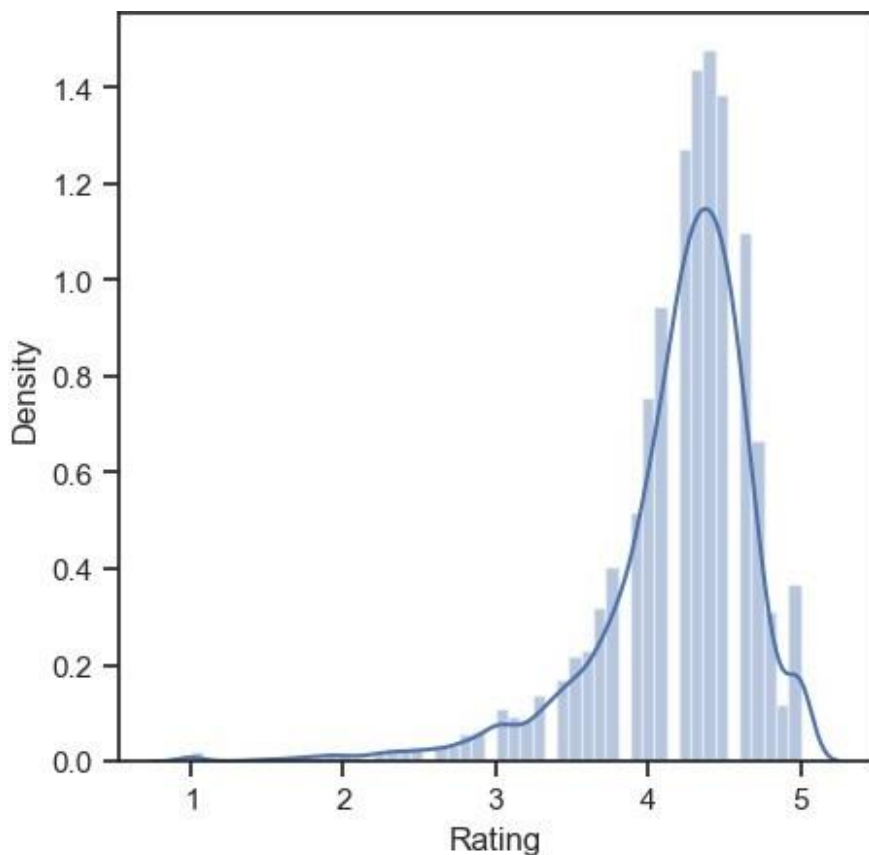
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(knnimpute_hdata['Rating'])
```

<Axes: xlabel='Rating', ylabel='Density'>



С помощью импьютации сохранили форму распределения, не создав пиков.

## кодирование признаков

### категориальные

```
data1 = pd.read_csv(r'C:\Users\ksarb\Documents\MMO_2024\Datasets\Life.csv', sep=",")
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
data1.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	\
0	Afghanistan	2015	Developing	65.0	263.0	
1	Afghanistan	2014	Developing	59.9	271.0	
2	Afghanistan	2013	Developing	59.9	268.0	
3	Afghanistan	2012	Developing	59.5	272.0	
4	Afghanistan	2011	Developing	59.2	275.0	

	infant deaths	Alcohol percentage	expenditure	Hepatitis B
Measles ... \				
0	62	0.01	71.279624	65.0

1154	...				
1		64	0.01	73.523582	62.0
492	...				
2		66	0.01	73.219243	64.0
430	...				
3		69	0.01	78.184215	67.0
2787	...				
4		71	0.01	7.097109	68.0
3013	...				

	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP
Population \					
0	6.0	8.16	65.0	0.1	584.259210
33736494.0					
1	58.0	8.18	62.0	0.1	612.696514
327582.0					
2	62.0	8.13	64.0	0.1	631.744976
31731688.0					
3	67.0	8.52	67.0	0.1	669.959000
3696958.0					
4	68.0	7.87	68.0	0.1	63.537231
2978599.0					

	thinness 1-19 years	thinness 5-9 years \
0	17.2	17.3
1	17.5	17.5
2	17.7	17.7
3	17.9	18.0
4	18.2	18.2

	Income composition of resources	Schooling
0	0.479	10.1
1	0.476	10.0
2	0.470	9.9
3	0.463	9.8
4	0.454	9.5

[5 rows x 22 columns]

```
data1['Country'].unique()
```

```
array(['Afghanistan', 'Albania', 'Algeria', 'Angola',
      'Antigua and Barbuda', 'Argentina', 'Armenia', 'Australia',
      'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh',
      'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
      'Bolivia (Plurinational State of)', 'Bosnia and Herzegovina',
      'Botswana', 'Brazil', 'Brunei Darussalam', 'Bulgaria',
      'Burkina Faso', 'Burundi', 'Côte d'Ivoire', 'Cabo Verde',
      'Cambodia', 'Cameroon', 'Canada', 'Central African Republic',
      'Chad', 'Chile', 'China', 'Colombia', 'Comoros', 'Congo',
```

'Cook Islands', 'Costa Rica', 'Croatia', 'Cuba', 'Cyprus',  
'Czechia', "Democratic People's Republic of Korea",  
'Democratic Republic of the Congo', 'Denmark', 'Djibouti',  
'Dominica', 'Dominican Republic', 'Ecuador', 'Egypt',  
'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',  
'Ethiopia', 'Fiji', 'Finland', 'France', 'Gabon', 'Gambia',  
'Georgia', 'Germany', 'Ghana', 'Greece', 'Grenada',  
'Guatemala',  
'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras',  
'Hungary', 'Iceland', 'India', 'Indonesia',  
'Iran (Islamic Republic of)', 'Iraq', 'Ireland', 'Israel',  
'Italy',  
'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya',  
'Kiribati',  
'Kuwait', 'Kyrgyzstan', "Lao People's Democratic Republic",  
'Latvia', 'Lebanon', 'Lesotho', 'Liberia', 'Libya',  
'Lithuania',  
'Luxembourg', 'Madagascar', 'Malawi', 'Malaysia', 'Maldives',  
'Mali', 'Malta', 'Marshall Islands', 'Mauritania', 'Mauritius',  
'Mexico', 'Micronesia (Federated States of)', 'Monaco',  
'Mongolia',  
'Montenegro', 'Morocco', 'Mozambique', 'Myanmar', 'Namibia',  
'Nauru', 'Nepal', 'Netherlands', 'New Zealand', 'Nicaragua',  
'Niger', 'Nigeria', 'Niue', 'Norway', 'Oman', 'Pakistan',  
'Palau',  
'Panama', 'Papua New Guinea', 'Paraguay', 'Peru',  
'Philippines',  
'Poland', 'Portugal', 'Qatar', 'Republic of Korea',  
'Republic of Moldova', 'Romania', 'Russian Federation',  
'Rwanda',  
'Saint Kitts and Nevis', 'Saint Lucia',  
'Saint Vincent and the Grenadines', 'Samoa', 'San Marino',  
'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',  
'Seychelles', 'Sierra Leone', 'Singapore', 'Slovakia',  
'Slovenia',  
'Solomon Islands', 'Somalia', 'South Africa', 'South Sudan',  
'Spain', 'Sri Lanka', 'Sudan', 'Suriname', 'Swaziland',  
'Sweden',  
'Switzerland', 'Syrian Arab Republic', 'Tajikistan',  
'Thailand',  
'The former Yugoslav republic of Macedonia', 'Timor-Leste',  
'Togo',  
'Tonga', 'Trinidad and Tobago', 'Tunisia', 'Turkey',  
'Turkmenistan', 'Tuvalu', 'Uganda', 'Ukraine',  
'United Arab Emirates',  
'United Kingdom of Great Britain and Northern Ireland',  
'United Republic of Tanzania', 'United States of America',  
'Uruguay', 'Uzbekistan', 'Vanuatu',

```

    'Venezuela (Bolivarian Republic of)', 'Viet Nam', 'Yemen',
    'Zambia', 'Zimbabwe'], dtype=object)

cat_enc_le = le.fit_transform(data1['Country'])

np.unique(cat_enc_le)

array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,
        13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
        25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
        38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
        64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,
        77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
        90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102,
        103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
        116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128,
        129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141,
        142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154,
        155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167,
        168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180,
        181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192])

le.inverse_transform([0, 1, 2, 3])

array(['Afghanistan', 'Albania', 'Algeria', 'Angola'], dtype=object)

pd.get_dummies(data1[['Country']]).head()

```

	Country_Afghanistan	Country_Albania	Country_Algeria	Country_Angola
0	True	False	False	False
1	True	False	False	False
2	True	False	False	False

3	True	False	False
False			
4	True	False	False
False			

Country_United Republic of Tanzania	Country_United States of America \
-------------------------------------	------------------------------------

0	False
False	
1	False
False	
2	False
False	
3	False
False	
4	False
False	

	Country_Yemen	Country_Zambia	Country_Zimbabwe
0	False	False	False
1	False	False	False
2	False	False	False



3	False	False	False
4	False	False	False

[5 rows x 193 columns]

## ЧИСЛОВЫЕ

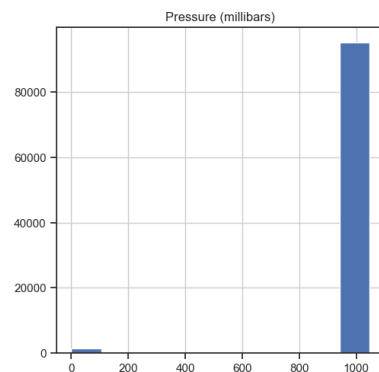
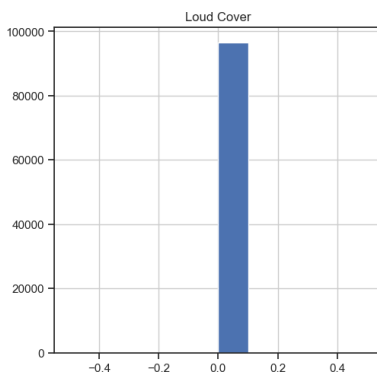
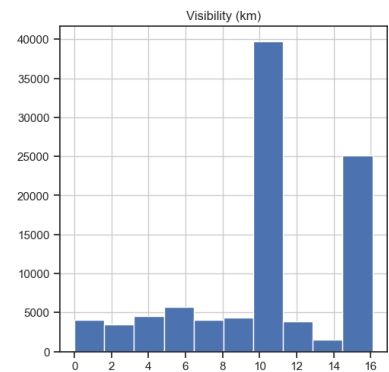
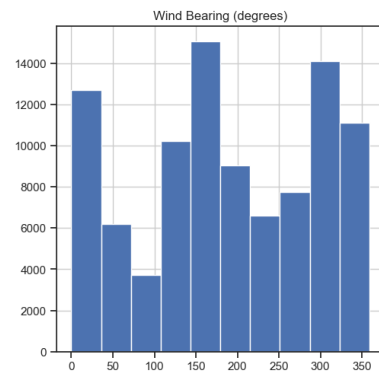
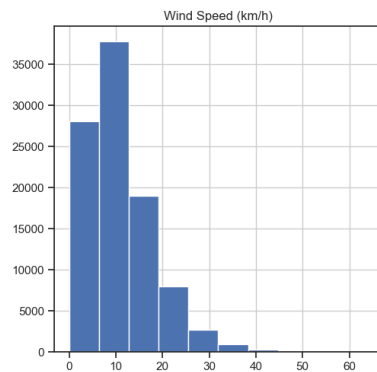
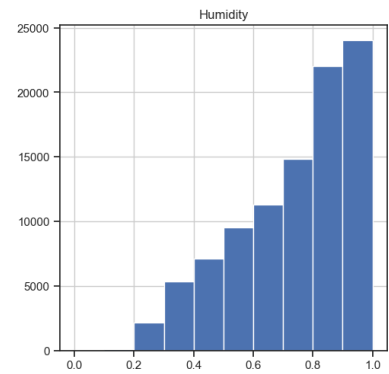
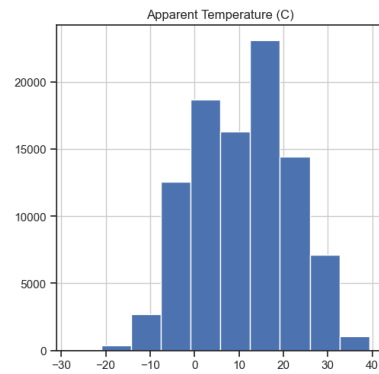
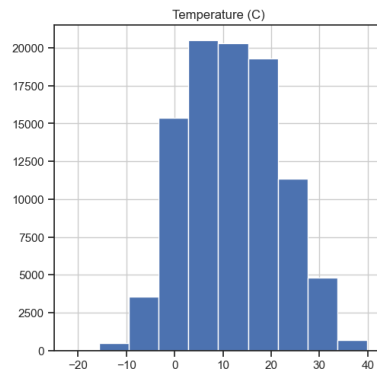
```
data2 = pd.read_csv(r'C:\Users\ksarb\Documents\MMO_2024\Datasets\weatherHistory.csv', sep=",")
```

```
def diagnostic_plots(df, variable):  
    plt.figure(figsize=(15,6))  
    # гистограмма  
    plt.subplot(1, 2, 1)  
    df[variable].hist(bins=30)  
    ## Q-Q plot  
    plt.subplot(1, 2, 2)  
    stats.probplot(df[variable], dist="norm", plot=plt)  
    plt.show()
```

```
data2.dtypes
```

Formatted Date	object
Summary	object
Precip Type	object
Temperature (C)	float64
Apparent Temperature (C)	float64
Humidity	float64
Wind Speed (km/h)	float64
Wind Bearing (degrees)	float64
Visibility (km)	float64
Loud Cover	float64
Pressure (millibars)	float64
Daily Summary	object
dtype:	object

```
data2.hist(figsize=(20,20))  
plt.show()
```



```
from sklearn.preprocessing import MinMaxScaler
# Обучаем StandardScaler на всей выборке и масштабируем
cs31 = MinMaxScaler()
data_cs31_scaled_temp = cs31.fit_transform(data2[['Apparent
Temperature (C)']])
# формируем DataFrame на основе массива
data_scaled = pd.DataFrame(data_cs31_scaled_temp, columns=['Apparent
Temperature (C)'])
data_scaled.describe()
```

	Apparent Temperature (C)
count	96453.000000
mean	0.575172

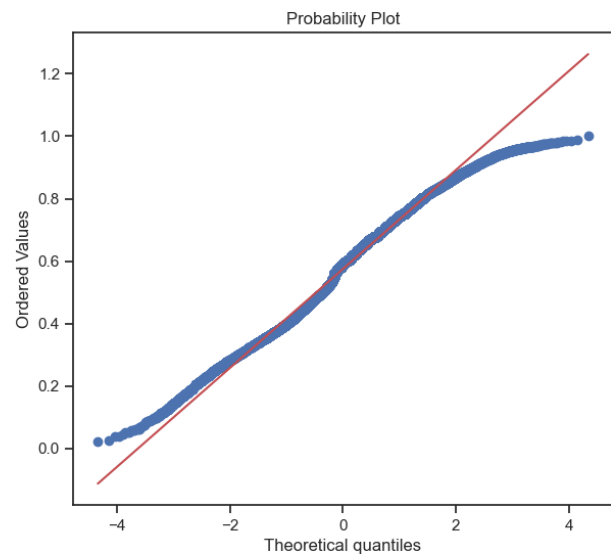
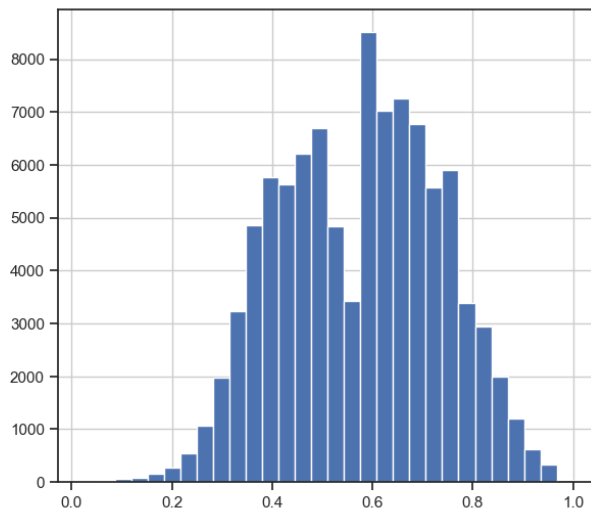
```
std      0.159509
min      0.000000
25%      0.447767
50%      0.592246
75%      0.694226
max      1.000000
```

```
data_scaled.loc[data_scaled['Apparent Temperature (C)']==0]
```

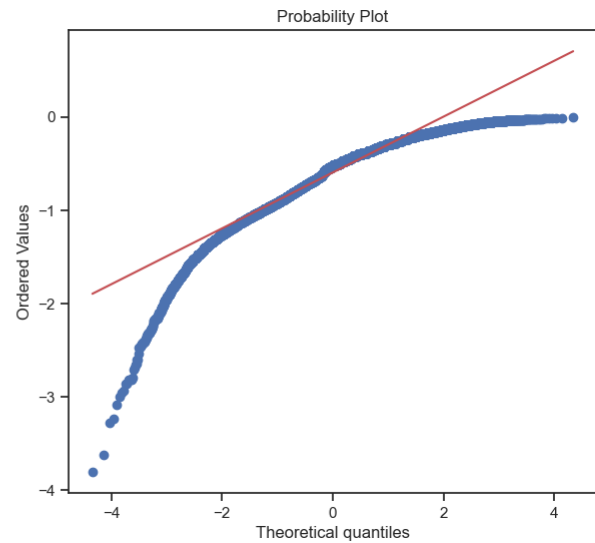
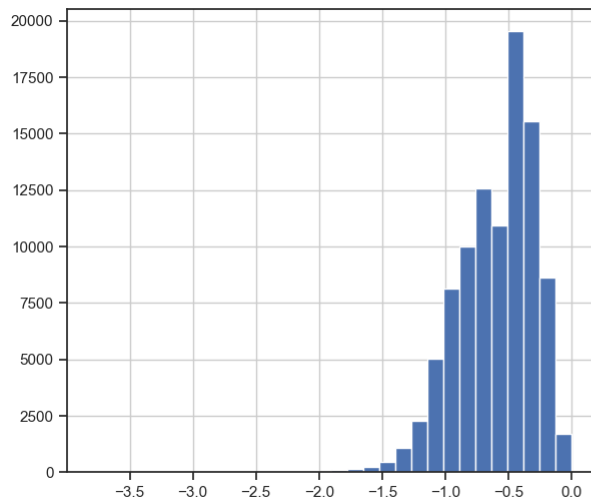
```
Apparent Temperature (C)
54864      0.0
```

```
data_scaled = data_scaled.loc[data_scaled['Apparent Temperature (C)']!=0]
```

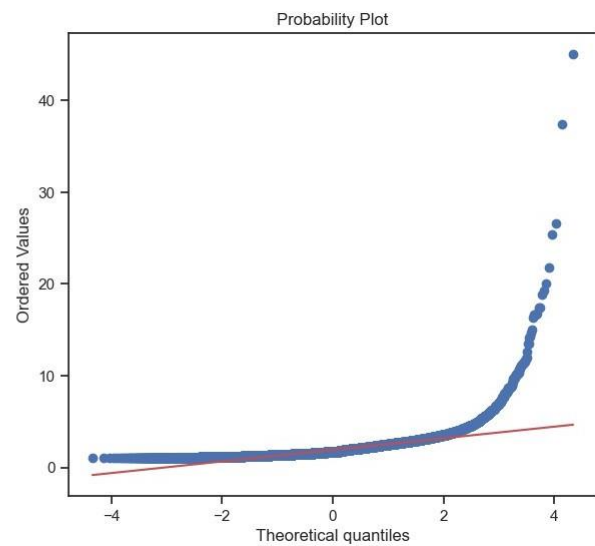
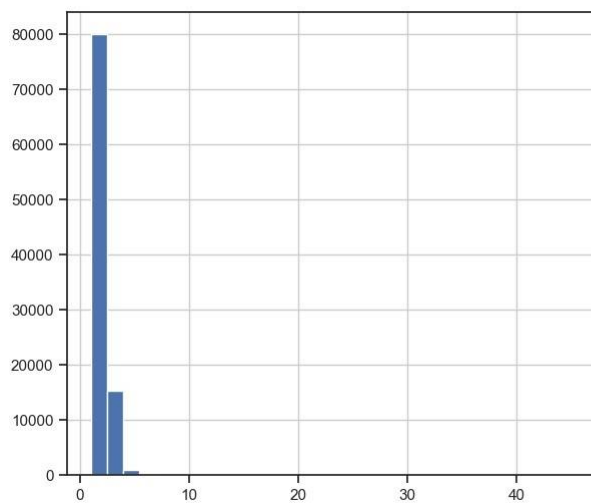
```
diagnostic_plots(data_scaled, 'Apparent Temperature (C)')
```



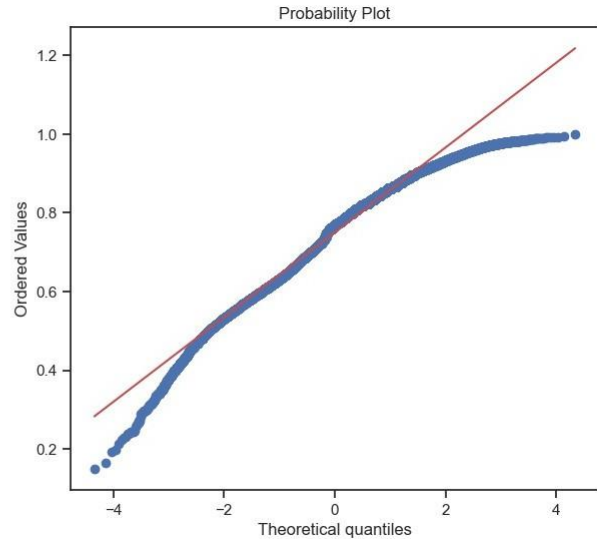
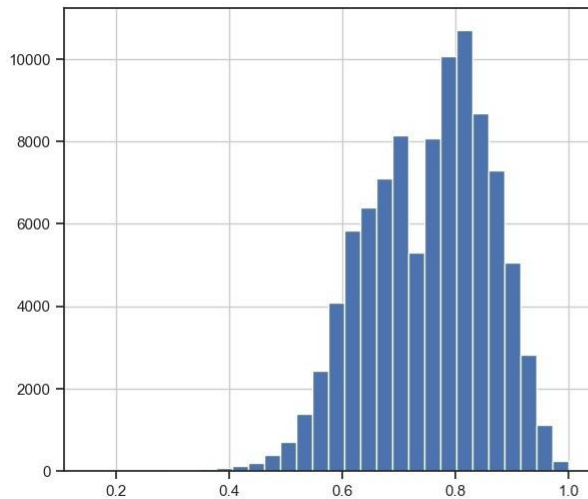
```
# логарифмическое
data_scaled['norm_log'] = np.log(data_scaled['Apparent Temperature (C)'])
diagnostic_plots(data_scaled, 'norm_log')
```



```
# обратное
data_scaled['norm_reciprocal'] = 1 / (data_scaled['Apparent
Temperature (C)'])
diagnostic_plots(data_scaled, 'norm_reciprocal')
```



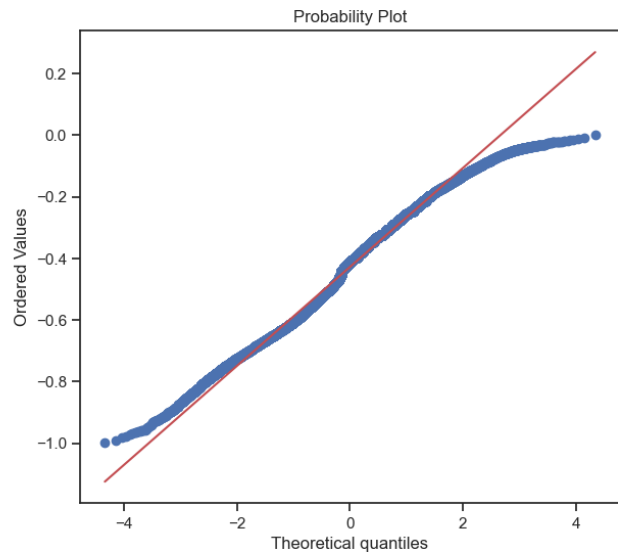
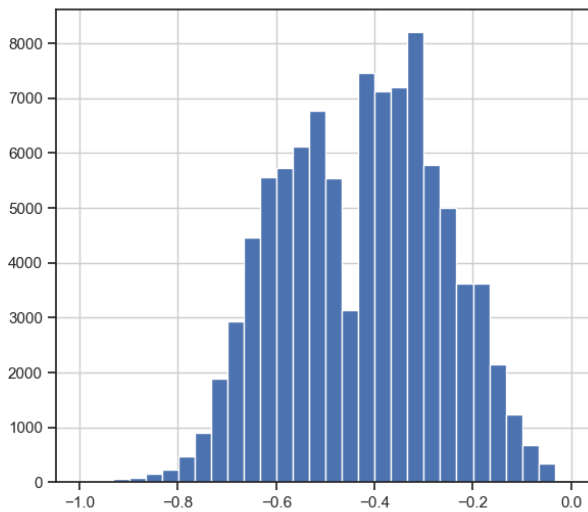
```
# root
data_scaled['norm_sqr'] = data_scaled['Apparent Temperature
(C)']**(1/2)
diagnostic_plots(data_scaled, 'norm_sqr')
```



*# Бокса-Кокса*

```
data_scaled['norm_boxcox'], param = stats.boxcox(data_scaled['Apparent
Temperature (C)'])
print('Оптимальное значение  $\lambda$  = {}'.format(param))
diagnostic_plots(data_scaled, 'norm_boxcox')
```

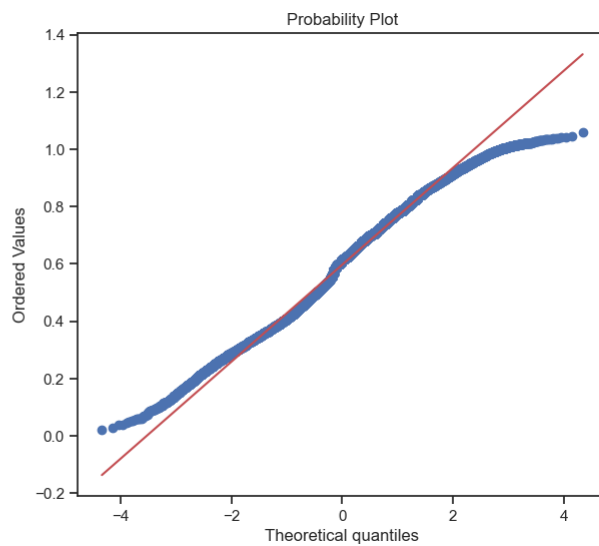
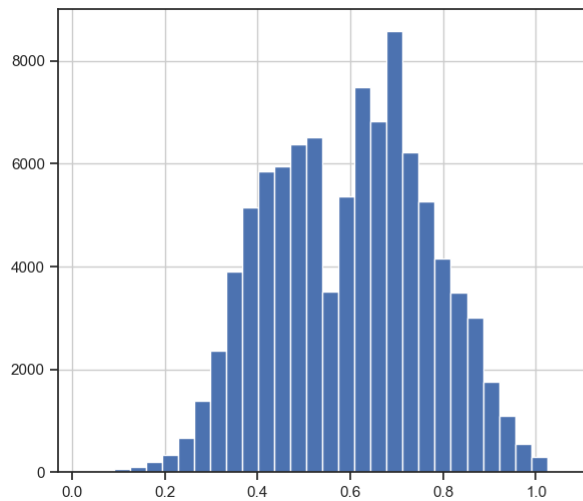
Оптимальное значение  $\lambda$  = 0.9778834045020082



*# Преобразование Йео-Джонсона*

```
data_scaled['norm_yeojohnson'], param =
stats.yeojohnson(data_scaled['Apparent Temperature (C)'])
print('Оптимальное значение  $\lambda$  = {}'.format(param))
diagnostic_plots(data_scaled, 'norm_yeojohnson')
```

Оптимальное значение  $\lambda$  = 1.1461032413095196



## **Вывод:**

Датасет необходимо подготавливать перед проведением любой работы по машинному обучению. В ЛР мы выполнили следующие стратегии работы с пропусками:

- Удаление строк с пропусками, где их количество менее 1% - они не являлись значимыми.
- Заполнение пропусков в rating – они составляют 10% от датасета. Были испробованы заполнение медианой – создало пик и нарушило распределение и импутация knn – сохранила распределение.

Категориальные признаки мы закодировали – чтобы была возможность сопоставить их в вычислениях с другими числовыми признаками, вычислить корреляцию.

Также была опробована нормализация – лучший результат дало преобразование Бокса-Кокса, однако оно потребовало предварительного масштабирования minmax до промежутка  $[0;1]$  с удалением 0 – чтобы стало  $(0;1]$