



Министерство науки и высшего образования  
Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
"Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)"  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА \_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)\_\_\_\_\_

## ОТЧЕТ

### Лабораторная работа №2

#### «Обработка признаков»

по курсу «Методы машинного обучения»

ИСПОЛНИТЕЛЬ:

группа ИУ5-21М

Савченко Г. А.

ФИО

\_\_\_\_\_

подпись

"\_\_" \_\_\_\_\_ 2023 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

ФИО

\_\_\_\_\_

подпись

"\_\_" \_\_\_\_\_ 2023 г.

---

Москва - 2023

# Лабораторная работа №2

## Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных.

Цель лабораторной работы: изучение способов предварительной обработки данных для дальнейшего формирования моделей.

### 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.

### Импорт библиотек

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

### Загрузка и первичный анализ данных

Используем данные с kaggle [Фильмы и телешоу Netflix](#)

```
In [2]: data = pd.read_csv('data/netflix_titles.csv', sep=",")
```

```
In [3]: # размер набора данных
data.shape
```

```
Out[3]: (8807, 12)
```

```
In [4]: # типы колонок
data.dtypes
```

```
Out[4]: show_id      object
type              object
title            object
director         object
```

```

cast          object
country       object
date_added    object
release_year   int64
rating        object
duration      object
listed_in     object
description    object
dtype: object

```

```

In [5]: # проверим есть ли пропущенные значения
data.isnull().sum()

```

```

Out[5]: show_id      0
type        0
title       0
director    2634
cast        825
country     831
date_added   10
release_year 0
rating       4
duration     3
listed_in    0
description  0
dtype: int64

```

```

In [6]: # Первые 5 строк датасета
data.head()

```

```

Out[6]:

```

	show_id	type	title	director	cast	country	date_added	release_year	rating	durati
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 m
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	Seasc
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Seas
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Seas
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra	India	September 24, 2021	2021	TV-MA	Seasc

show_id	type	title	director	cast	country	date_added	release_year	rating	durati
				Kumar, Ranjan Raj, Alam K...					

```
In [7]: total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

Всего строк: 8807

## Обработка пропусков в данных

```
In [8]: # Выберем колонки с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count > 0:
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

Колонка director. Тип данных object. Количество пустых значений 2634, 29.91%.  
Колонка cast. Тип данных object. Количество пустых значений 825, 9.37%.  
Колонка country. Тип данных object. Количество пустых значений 831, 9.44%.  
Колонка date\_added. Тип данных object. Количество пустых значений 10, 0.11%.  
Колонка rating. Тип данных object. Количество пустых значений 4, 0.05%.  
Колонка duration. Тип данных object. Количество пустых значений 3, 0.03%.

```
In [9]: # Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num
```

```
Out[9]:
```

	director	cast	country	date_added	rating	duration
0	Kirsten Johnson	NaN	United States	September 25, 2021	PG-13	90 min
1	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	TV-MA	2 Seasons
2	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	TV-MA	1 Season
3	NaN	NaN	NaN	September 24, 2021	TV-MA	1 Season
4	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	TV-MA	2 Seasons
...	...	...	...	...	...	...
8802	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	R	158 min
8803	NaN	NaN	NaN	July 1, 2019	TV-Y7	2 Seasons

	director	cast	country	date_added	rating	duration
<b>8804</b>	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	R	88 min
<b>8805</b>	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	PG	88 min
<b>8806</b>	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	TV-14	111 min

8807 rows × 6 columns

```
In [10]: cat_temp_data = data[['rating']]
cat_temp_data.head()
```

```
Out[10]: rating
0  PG-13
1  TV-MA
2  TV-MA
3  TV-MA
4  TV-MA
```

```
In [11]: cat_temp_data['rating'].unique()
```

```
Out[11]: array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
        'TV-G', 'G', 'NC-17', '74 min', '84 min', '66 min', 'NR', nan,
        'TV-Y7-FV', 'UR'], dtype=object)
```

```
In [12]: cat_temp_data[cat_temp_data['rating'].isnull()].shape
```

```
Out[12]: (4, 1)
```

Подключим библиотеки для обработки пропусков

```
In [13]: from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

```
In [14]: # Импутация наиболее частыми значениями
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

```
Out[14]: array(['PG-13',
        'TV-MA',
        'TV-MA',
        ...,
        'R',
        'PG',
        'TV-14'], dtype=object)
```

```
In [15]: # Пустые значения отсутствуют
```

```
np.unique(data_imp2)
```

```
Out[15]: array(['66 min', '74 min', '84 min', 'G', 'NC-17', 'NR', 'PG', 'PG-13',  
        'R', 'TV-14', 'TV-G', 'TV-MA', 'TV-PG', 'TV-Y', 'TV-Y7',  
        'TV-Y7-FV', 'UR'], dtype=object)
```

```
In [16]: # Импутация константой  
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='NA')  
data_imp3 = imp3.fit_transform(cat_temp_data)  
data_imp3
```

```
Out[16]: array([[ 'PG-13'],  
        [ 'TV-MA'],  
        [ 'TV-MA'],  
        ...,  
        [ 'R'],  
        [ 'PG'],  
        [ 'TV-14']], dtype=object)
```

```
In [17]: np.unique(data_imp3)
```

```
Out[17]: array(['66 min', '74 min', '84 min', 'G', 'NA', 'NC-17', 'NR', 'PG',  
        'PG-13', 'R', 'TV-14', 'TV-G', 'TV-MA', 'TV-PG', 'TV-Y', 'TV-Y7',  
        'TV-Y7-FV', 'UR'], dtype=object)
```

```
In [18]: data_imp3[data_imp3=='NA'].size
```

```
Out[18]: 4
```

## Кодирование категориальных признаков

### Кодирование категорий целочисленными значениями - **label encoding**

```
In [19]: cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})  
cat_enc
```

```
Out[19]:
```

	c1
0	PG-13
1	TV-MA
2	TV-MA
3	TV-MA
4	TV-MA
...	...
8802	R
8803	TV-Y7
8804	R
8805	PG
8806	TV-14

8807 rows × 1 columns

```
In [20]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
In [21]: le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
```

```
In [22]: np.unique(cat_enc_le)
```

```
Out[22]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

```
In [23]: le.inverse_transform([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13])
```

```
Out[23]: array(['66 min', '74 min', '84 min', 'G', 'NC-17', 'NR', 'PG', 'PG-13',
        'R', 'TV-14', 'TV-G', 'TV-MA', 'TV-PG', 'TV-Y'], dtype=object)
```

## Кодирование категорий наборами бинарных значений - **one-hot encoding**

```
In [24]: ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])
```

```
In [25]: cat_enc.shape
```

```
Out[25]: (8807, 1)
```

```
In [26]: cat_enc_ohe.shape
```

```
Out[26]: (8807, 17)
```

```
In [27]: cat_enc_ohe.todense()[0:10]
```

```
Out[27]: matrix([[0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,  
    0.],  
    [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,  
    0.],  
    [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,  
    0.],  
    [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,  
    0.],  
    [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,  
    0.],  
    [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,  
    0.],  
    [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,  
    0.],  
    [0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
    0.],  
    [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,  
    0.],  
    [0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,  
    0.]])
```

```
In [28]: cat_enc.head(10)
```

```
Out[28]:
```

	c1
0	PG-13
1	TV-MA
2	TV-MA
3	TV-MA
4	TV-MA
5	TV-MA
6	PG
7	TV-MA
8	TV-14
9	PG-13

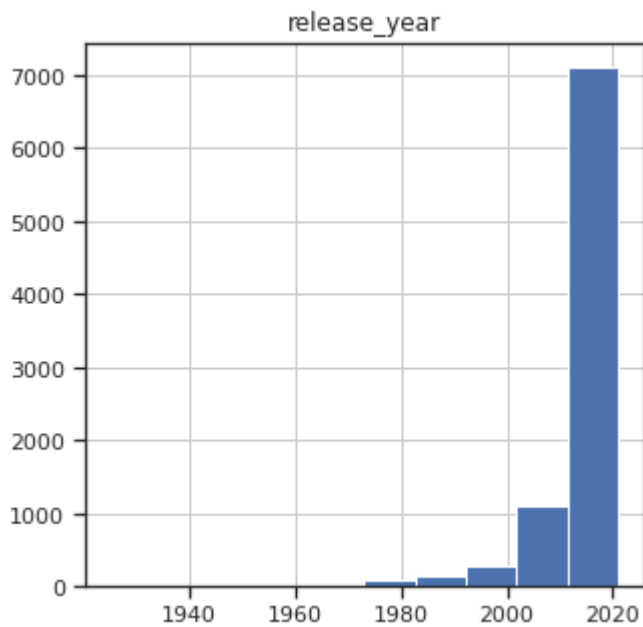
## Нормализация числовых признаков

```
In [34]: import scipy.stats as stats
```

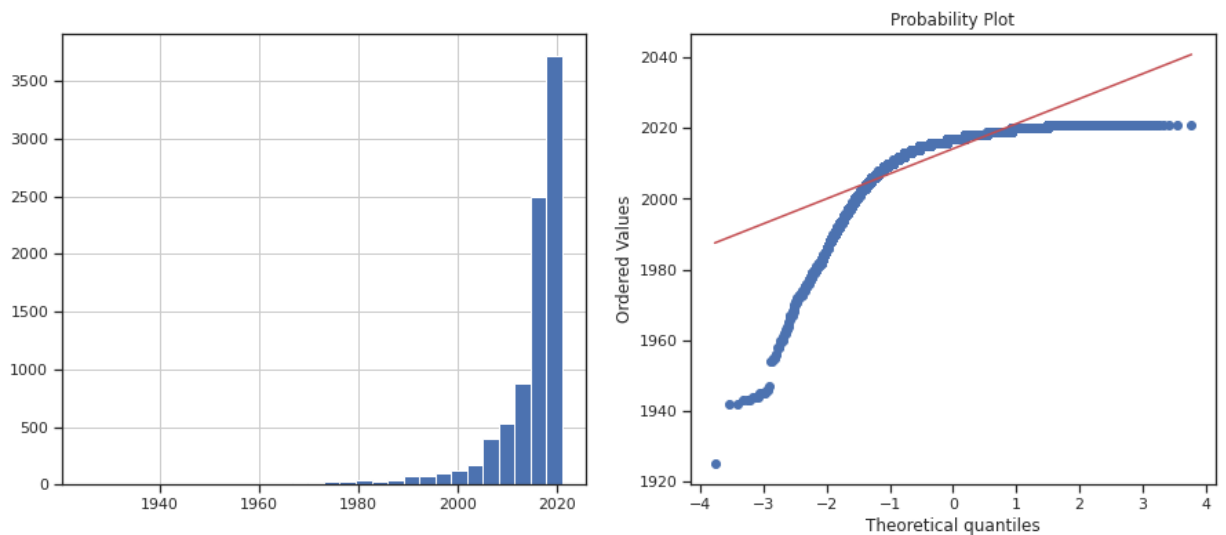
```
In [35]: def diagnostic_plots(df, variable):  
    plt.figure(figsize=(15,6))  
    # zусмотрамма  
    plt.subplot(1, 2, 1)  
    df[variable].hist(bins=30)  
    ## Q-Q plot  
    plt.subplot(1, 2, 2)  
    stats.probplot(df[variable], dist="norm", plot=plt)  
    plt.show()
```

```
In [30]: data.hist(figsize=(5,5))  
plt.show()
```



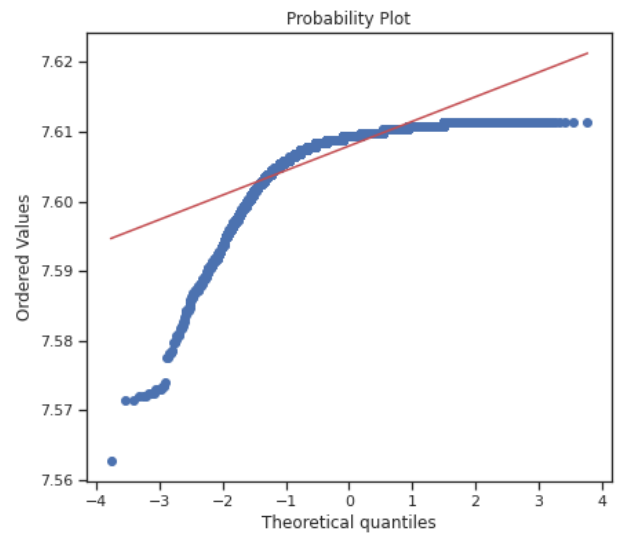
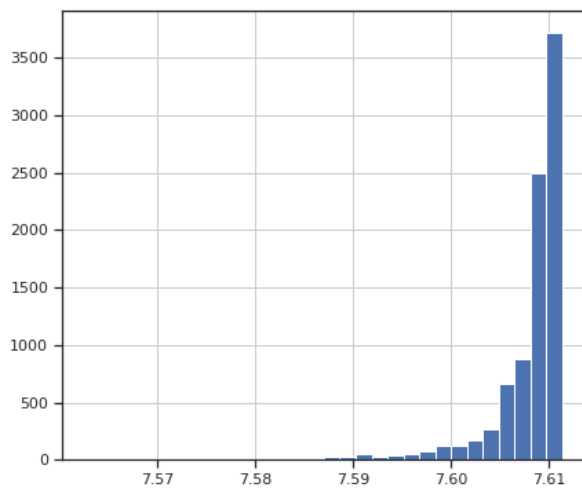


```
In [36]: diagnostic_plots(data, 'release_year')
```



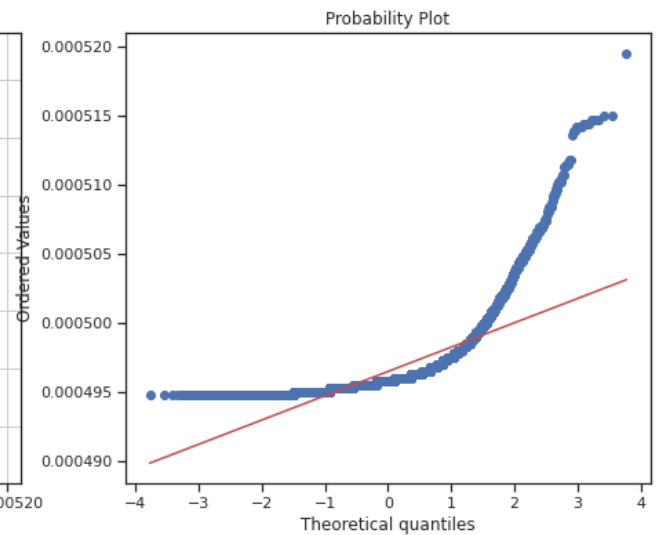
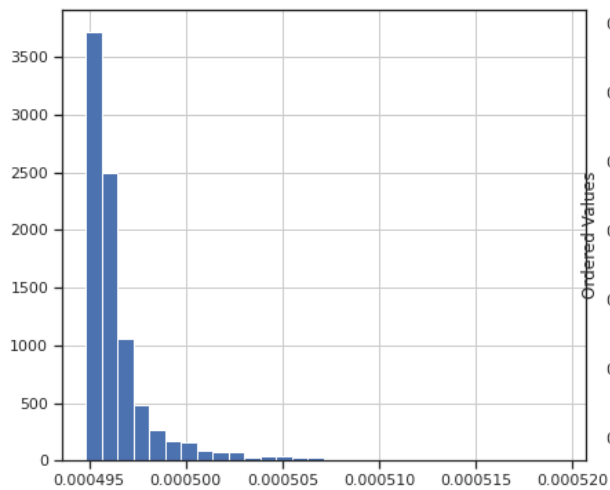
## Логарифмическое преобразование

```
In [38]: data['release_year_log'] = np.log(data['release_year'])
diagnostic_plots(data, 'release_year_log')
```



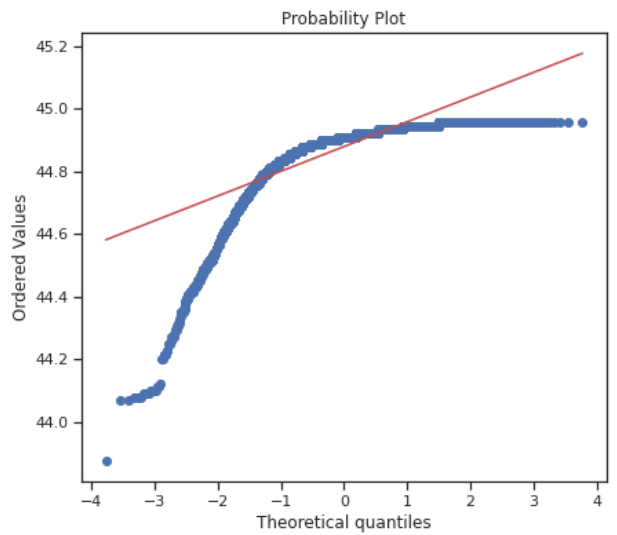
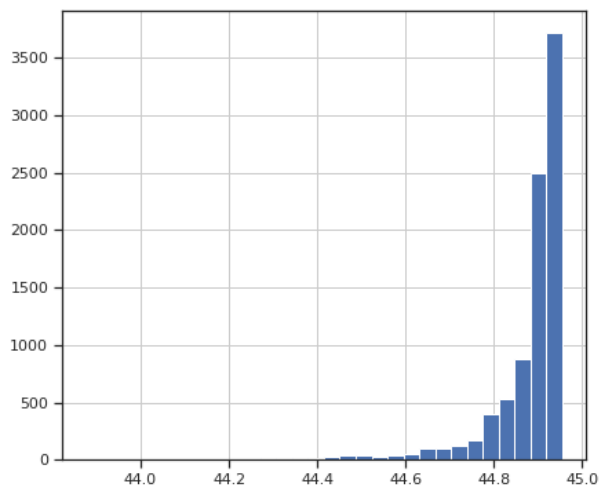
## Обратное преобразование

```
In [39]: data['release_year_reciprocal'] = 1 / (data['release_year'])
          diagnostic_plots(data, 'release_year_reciprocal')
```



## Квадратный корень

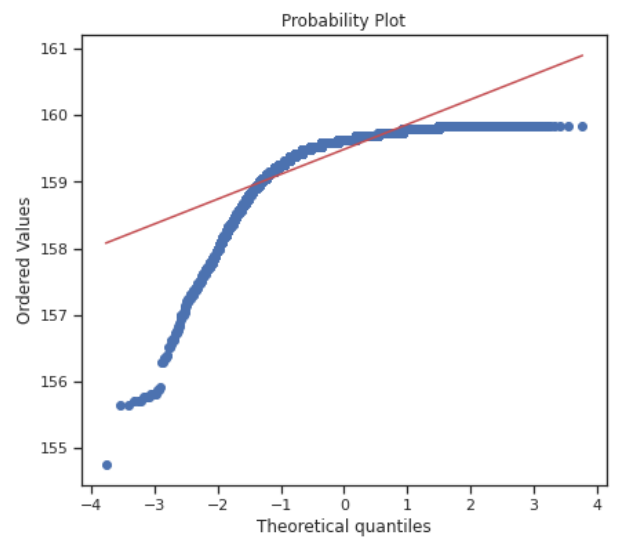
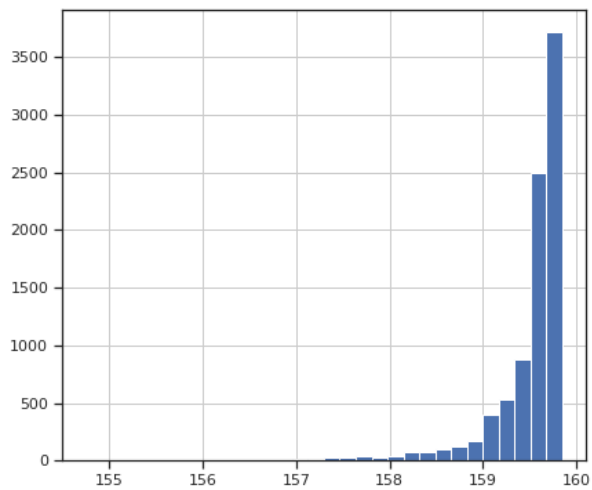
```
In [40]: data['release_year_sqr'] = data['release_year']**(1/2)
          diagnostic_plots(data, 'release_year_sqr')
```



## Возведение в степень

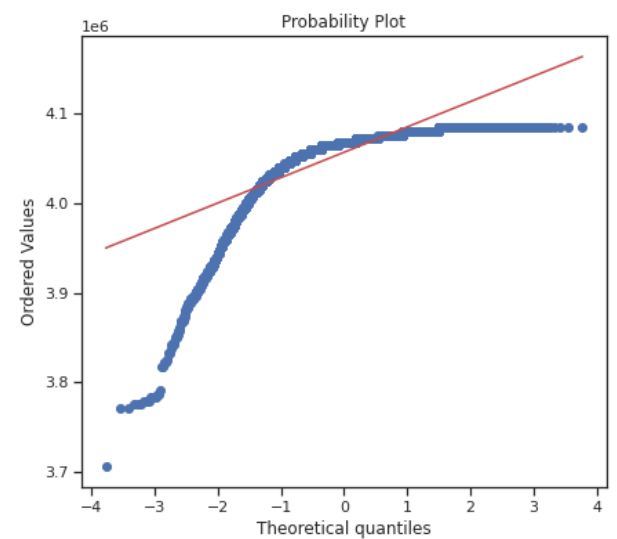
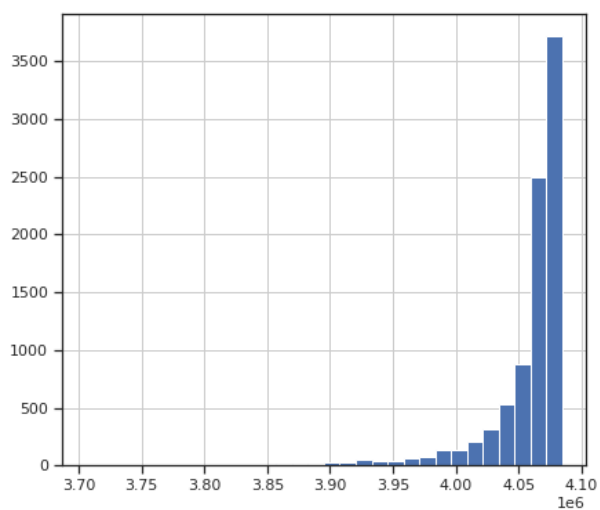
In [41]:

```
data['release_year_exp1'] = data['release_year']**(1/1.5)
diagnostic_plots(data, 'release_year_exp1')
```

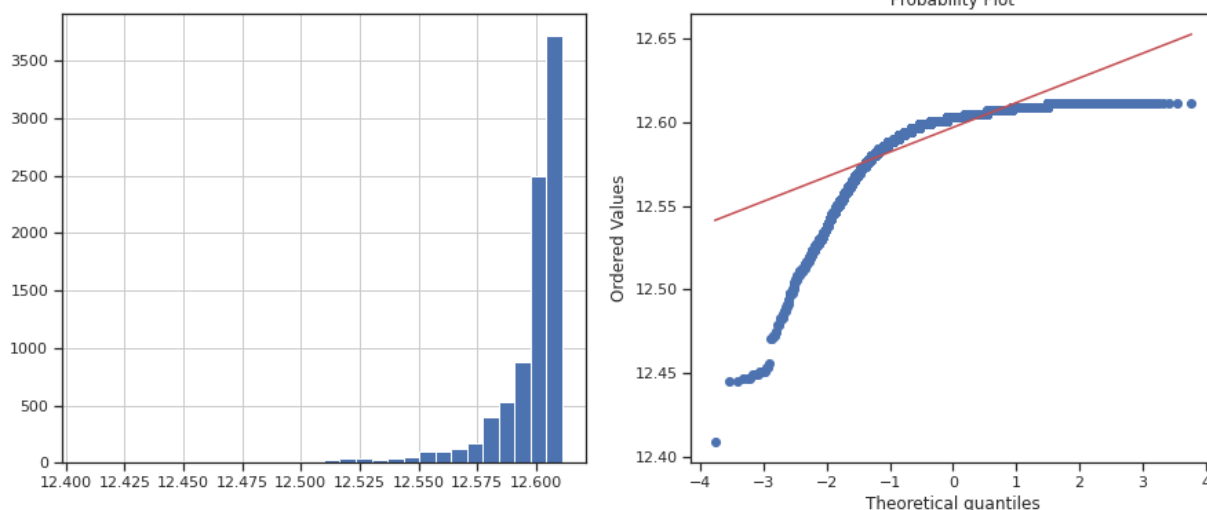


In [42]:

```
data['release_year_exp2'] = data['release_year']**(2)
diagnostic_plots(data, 'release_year_exp2')
```



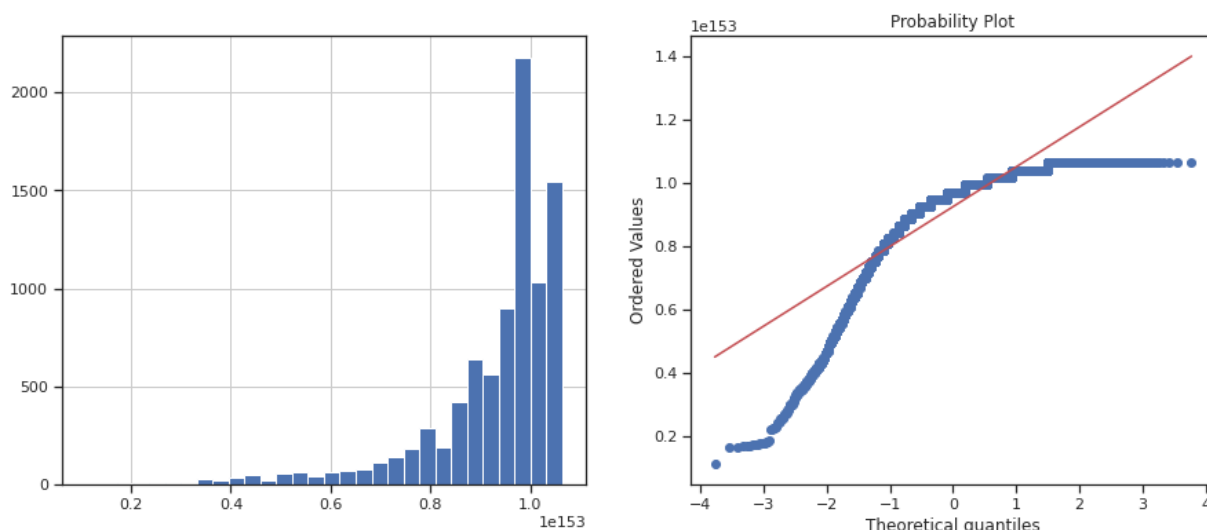
```
In [43]: data['release_year_exp3'] = data['release_year']**(0.333)
diagnostic_plots(data, 'release_year_exp3')
```



## Преобразование Бокса-Кокса

```
In [44]: data['release_year_boxcox'], param = stats.boxcox(data['release_year'])
print('Оптимальное значение  $\lambda$  = {}'.format(param))
diagnostic_plots(data, 'release_year_boxcox')
```

/root/miniconda3/lib/python3.9/site-packages/numpy/core/\_methods.py:205: RuntimeWarning: overflow encountered in multiply  
 x = um.multiply(x, x, out=x)  
/root/miniconda3/lib/python3.9/site-packages/numpy/core/\_methods.py:216: RuntimeWarning: overflow encountered in reduce  
 ret = umr\_sum(x, axis, dtype, out, keepdims)  
Оптимальное значение  $\lambda$  = 46.79897643237659



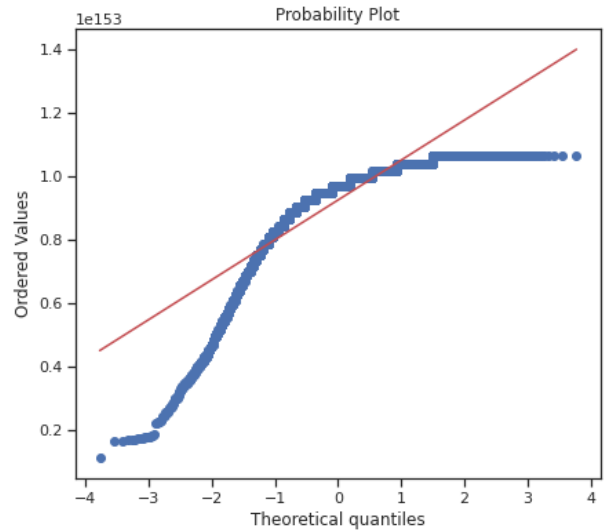
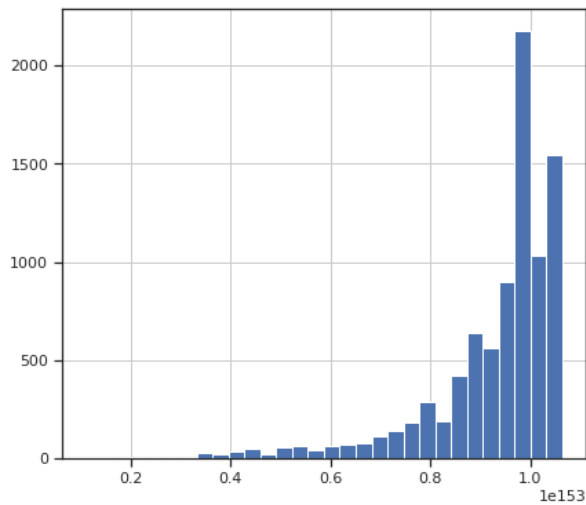
## Преобразование Йео-Джонсона

```
In [45]: # Необходимо преобразовать данные к действительному типу
data['release_year'] = data['release_year'].astype('float')
data['release_year_yeojohnson'], param = stats.yeojohnson(data['release_year'])
print('Оптимальное значение  $\lambda$  = {}'.format(param))
diagnostic_plots(data, 'release_year_yeojohnson')
```

```

/root/miniconda3/lib/python3.9/site-packages/numpy/core/_methods.py:205: RuntimeWarn
ing: overflow encountered in multiply
  x = um.multiply(x, x, out=x)
/root/miniconda3/lib/python3.9/site-packages/numpy/core/_methods.py:216: RuntimeWarn
ing: overflow encountered in reduce
  ret = umr_sum(x, axis, dtype, out, keepdims)
Оптимальное значение  $\lambda = 46.795973629169694$ 

```



## Масштабирование данных

```

In [37]: from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

```

### MinMax масштабирование

```

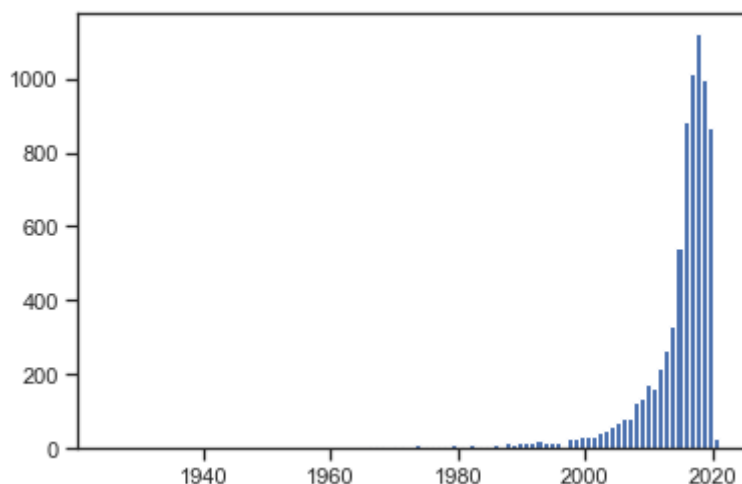
In [38]: sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['release_year']])

```

```

In [39]: plt.hist(data['release_year'], 100)
plt.show()

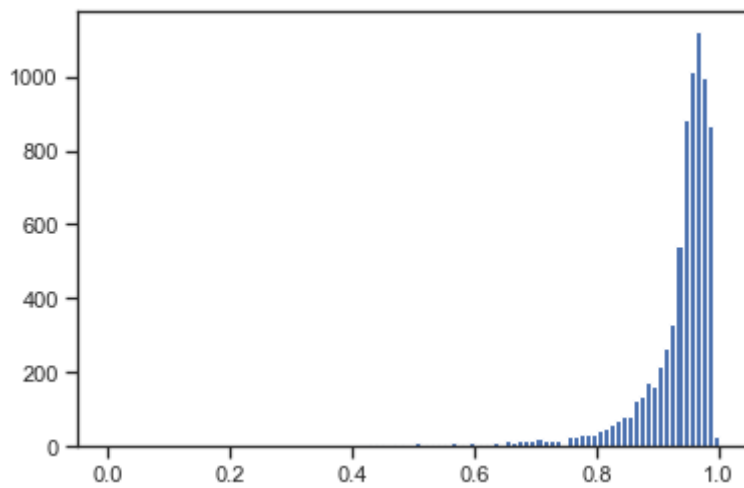
```



```

In [40]: plt.hist(sc1_data, 100)
plt.show()

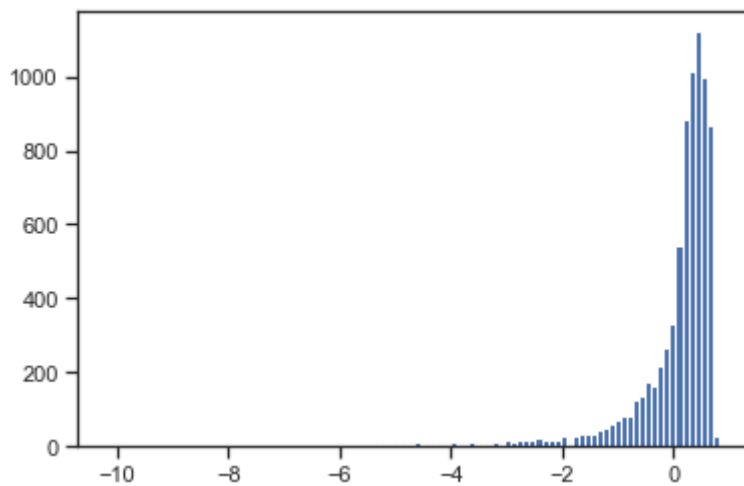
```



## Масштабирование данных на основе Z-оценки - **StandardScaler**

```
In [41]: sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['release_year']])
```

```
In [42]: plt.hist(sc2_data, 100)
plt.show()
```



In [ ]:

In [ ]: