

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

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

ОТЧЕТ

Лабораторная работа № 2
по дисциплине «Методы машинного обучения в автоматизированных
системах обработки информации и управления»

ИСПОЛНИТЕЛЬ:

группа ИУ5-25М

Стрихар П.А.

ФИО

подпись

"23" апреля 2024 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

ФИО

подпись

" " 2024 г.

Москва – 2024

✓ Задание лабораторной работы

- 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.
- Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
 - устранение пропусков в данных;
 - кодирование категориальных признаков;
 - нормализация числовых признаков.


✓ Выполнение работы

✓ Импорт библиотек

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from IPython.display import Image
%matplotlib inline
sns.set(style="ticks")
```

✓ Подключение Google Диска для работы с Google Colab

```
from google.colab import drive
drive.mount('/content/drive')
```

 Mounted at /content/drive


✓ Чтение данных

```
data = pd.read_csv('/content/drive/MyDrive/netflix_titles.csv', encoding='unicode_escape')
```

```
data.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalanane, Thabani...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
					Sami Rouaiiila						Crime TV	

```
data.shape
```

 (8807, 12)

✓ Устранение пропусков

Определим столбцы, в которых наблюдаются пропуски данных:

```
for column in data.columns:
    if (data[column].isnull().sum() != 0):
        print(column, ': ', data[column].isnull().sum())
```

```
director : 2634
cast : 825
country : 831
date_added : 10
rating : 4
duration : 3
```

Удалим пропуски в director:

```
data.drop(data[data['director'].isnull()].index, inplace=True)
```

Проверим снова:

```
for column in data.columns:
    if (data[column].isnull().sum() != 0):
        print(column, ': ', data[column].isnull().sum())
```

```
cast : 473
country : 422
rating : 1
duration : 3
```

Удалим еще строки с пропусками:

```
data.drop(data[data['cast'].isnull()].index, inplace=True)
data.drop(data[data['country'].isnull()].index, inplace=True)
data.drop(data[data['rating'].isnull()].index, inplace=True)
data.drop(data[data['duration'].isnull()].index, inplace=True)
```

Убедимся с помощью процентного соотношения, что пропусков в столбцах нет:

```
for col in data.columns:
    pct_missing = np.mean(data[col].isnull())
    print('{} - {}'.format(col, round(pct_missing*100)))
```

```
show_id - 0%
type - 0%
title - 0%
director - 0%
cast - 0%
country - 0%
date_added - 0%
release_year - 0%
rating - 0%
duration - 0%
listed_in - 0%
description - 0%
```

✓ Кодирование категориальных признаков

✓ LabelEncoder

Выберем два категориальных признака - type и director. Их закодируем с помощью LabelEncoder.

```
data.type.unique()
```

```
array(['Movie', 'TV Show'], dtype=object)
```

```
data.director.unique()
```

```
array(['Haile Gerima', 'Andy Devonshire', 'Theodore Melfi', ...,
      'Majid Al Ansari', 'Peter Hewitt', 'Mozes Singh'], dtype=object)
```

Закодируем их в числовые значения:

```
from sklearn.preprocessing import LabelEncoder
```

```
letype = LabelEncoder()
learrtype = letype.fit_transform(data["type"])
data["type"] = learrtype
data = data.astype({"type": "int64"})
```

```
lepriv = LabelEncoder()
learrpriv = lepriv.fit_transform(data["director"])
data["director"] = learrpriv
data = data.astype({"director": "int64"})
```

```
data.type.unique()
```

```
array([0, 1])
```

```
data.director.unique()
```

```
array([1309, 256, 3619, ..., 2201, 2817, 2537])
```

✓ OneHotEncoder

Для признака rating проведем кодирование бинарными значениями с помощью OneHotEncoder.

```
data.rating.unique()
```

```
array(['TV-MA', 'TV-14', 'PG-13', 'PG', 'R', 'TV-PG', 'G', 'TV-Y7',
      'TV-G', 'TV-Y', 'NC-17', 'NR', 'TV-Y7-FV', 'UR'], dtype=object)
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(data[['rating']])
cat_enc_ohe
```

```
<5332x14 sparse matrix of type '<class 'numpy.float64'>'
with 5332 stored elements in Compressed Sparse Row format>
```

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

```
matrix([[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., 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., 1., 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.],
        [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.]])
```

```
pd.get_dummies(data[['rating']]).head()
```

```
rating_G rating_NC-17 rating_NR rating_PG rating_PG-13 rating_R rating_TV-14 rat
7 False False False False False False False
8 False False False False False False True
9 False False False False True False False
12 False False False False False False False
```

```
pd.get_dummies(data[['rating']], dummy_na=True).head()
```

	rating_G	rating_NC-17	rating_NR	rating_PG	rating_PG-13	rating_R	rating_TV-14	rat
7	False	False	False	False	False	False	False	
8	False	False	False	False	False	False	True	
9	False	False	False	False	True	False	False	
12	False	False	False	False	False	False	False	

CountEncoder

Для кодирования country используем CountEncoder.

```
data.country.unique()
```

```
array(['United States, Ghana, Burkina Faso, United Kingdom, Germany, Ethiopia',
      'United Kingdom', 'United States', 'Germany, Czech Republic',
      'India', 'United States, India, France',
      'China, Canada, United States',
      'South Africa, United States, Japan', 'Japan', 'Nigeria',
      'Spain, United States', 'United Kingdom, United States',
      'United Kingdom, Australia, France',
      'United Kingdom, Australia, France, United States',
      'United States, Canada', 'Germany, United States',
      'South Africa, United States', 'United States, Mexico',
      'United States, Italy, France, Japan',
      'United States, Italy, Romania, United Kingdom',
      'Australia, United States', 'Argentina, Venezuela',
      'United States, United Kingdom, Canada', 'China, Hong Kong',
      'Canada', 'Hong Kong', 'United States, China, Hong Kong',
      'Italy, United States', 'United States, Germany', 'France',
      'United Kingdom, Canada, United States',
      'United States, United Kingdom', 'India, Nepal',
      'New Zealand, Australia, France, United States',
      'Italy, Brazil, Greece', 'Spain', 'Colombia',
      'United States, Japan', 'Mexico',
      'Switzerland, United Kingdom, Australia', 'South Africa',
      'Canada, United States', 'Argentina', 'Argentina, Spain',
      'United States, Nigeria', 'Taiwan', 'Bulgaria, United States',
      'Spain, United Kingdom, United States', 'United States, China',
      'United States, France',
      'Spain, France, United Kingdom, United States',
      'France, Algeria', 'Poland',
      'France, Israel, Germany, United States, United Kingdom',
      'Australia', 'New Zealand', 'Saudi Arabia', 'Thailand',
      'Indonesia', 'Italy', 'United States, Switzerland',
      'Hong Kong, Canada, United States', 'Kuwait, United States',
      'France, Canada, United States, Spain',
      'France, Netherlands, Singapore', 'Egypt', 'Malaysia',
      'South Korea, Czech Republic', 'South Korea', 'Vietnam',
      'United Kingdom, Belgium',
      'United Kingdom, Australia, United States',
      'France, Japan, United States',
      'United Kingdom, Germany, Spain, United States',
      'United Kingdom, United States, France, Italy',
      'United States, Germany, Canada',
      'United States, France, Italy, United Kingdom',
      'United States, United Kingdom, Germany, Hungary',
      'United States, New Zealand', 'Lebanon', 'Brazil', 'Romania',
      'Lebanon, Syria', 'Philippines', 'Germany', 'United States, India',
      'Philippines, Singapore, Indonesia',
      'China, United States, Canada', 'Lebanon, United Arab Emirates',
      'Canada, United States, Denmark', 'United Arab Emirates',
      'Mexico, France, Colombia', 'Sweden',
      'Germany, United States, France', 'United States, Bulgaria',
      'United Kingdom, France, Germany, United States',
      'Syria, France, Lebanon, Qatar', 'Belgium, Netherlands',
      'Mauritius', 'Canada, South Africa', 'Austria', 'Mexico, Brazil',
      'France, United States', 'China', 'Turkey', 'Germany, France',
      'United Kingdom, France, Spain, United States',
      'United States, Australia',
      'United States, United Kingdom, France', 'United States, Russia',
      'United States, United Kingdom, New Zealand',
```

```
!pip install category_encoders
from category_encoders.count import CountEncoder as ce_CountEncoder
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
    81.9/81.9 kB 853.4 kB/s eta 0:00:00
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.25.2)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.2.2)
```

```
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.11.4)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.14.2)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (2.0.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.6)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2022.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2022.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2022.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encoders) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encoders) (3.1.0)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->category_encoders) (23.1)
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.3
```

```
ce_CountEncoder1 = ce_CountEncoder()
data_COUNT_ENC = ce_CountEncoder1.fit_transform(data['country'])
```

```
data_COUNT_ENC.country.unique()
```

```
array([[ 1, 183, 1846, 875, 3, 83, 88, 59, 52, 12, 4,
        6, 8, 2, 13, 107, 49, 16, 73, 5, 35, 91,
        64, 23, 29, 7, 19, 10, 41, 76, 90, 48, 9,
        46, 74, 79, 14, 24, 11, 15])
```

✓ FrequencyEncoder

Для признака `type` используем `FrequencyEncoder`.

```
data.type.unique()
```

```
array(['Movie', 'TV Show'], dtype=object)
```

```
ce_CountEncoder2 = ce_CountEncoder(normalize=True)
data_FREQ_ENC = ce_CountEncoder2.fit_transform(data['type'])
```

```
data_FREQ_ENC.type.unique()
```

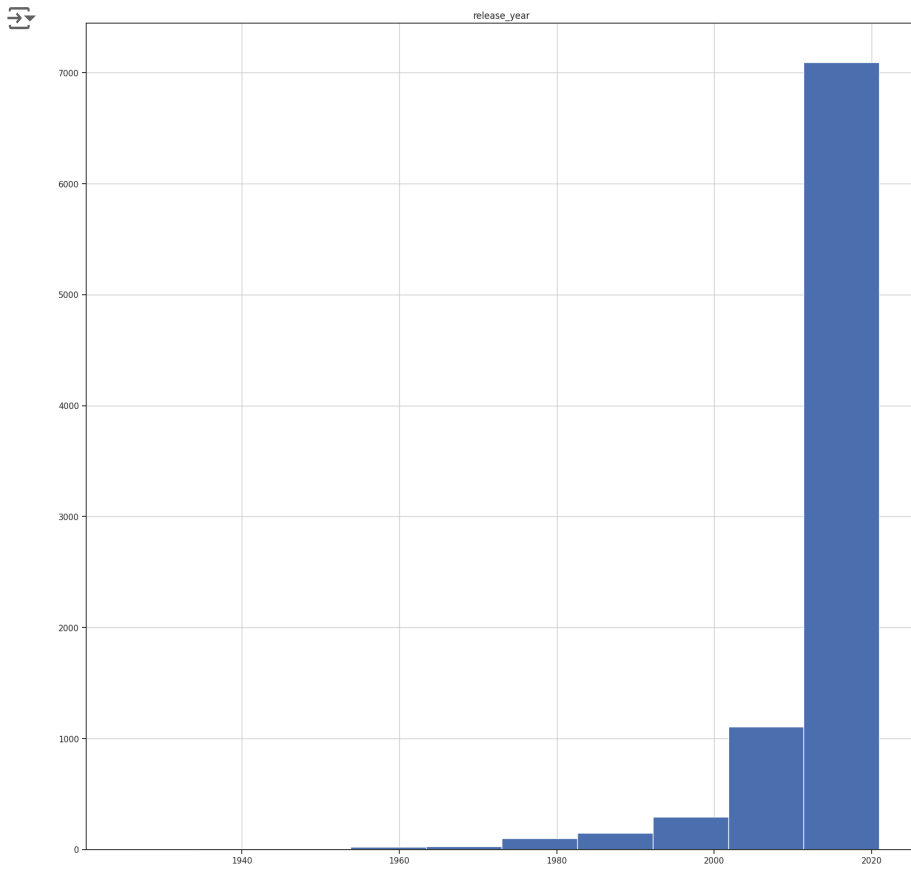
```
array([0.69615079, 0.30384921])
```

✓ Нормализация числовых признаков

Нормализация числового признака предполагает что на основе существующего признака мы создаем новый признак, который в идеале имеет нормальное распределение.

```
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()
```

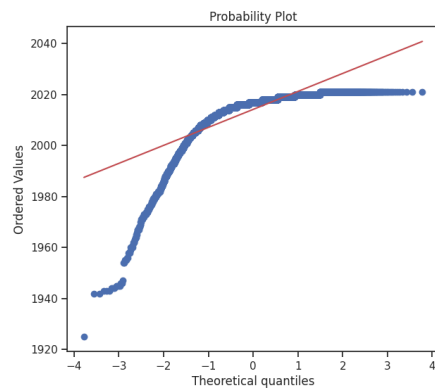
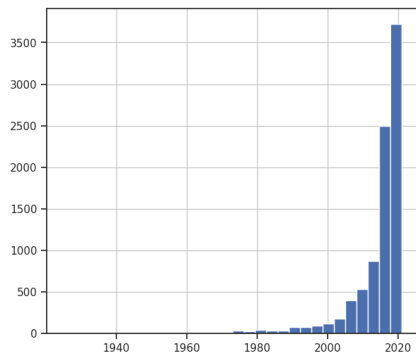
```
data.hist(figsize=(20,20))
plt.show()
```



✓ Исходное распределение

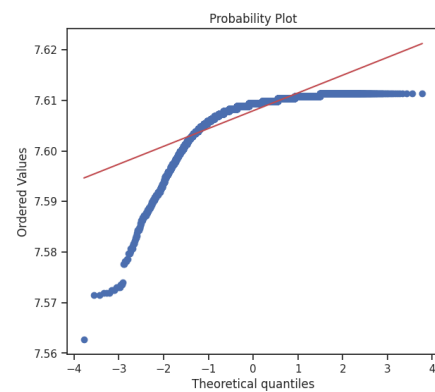
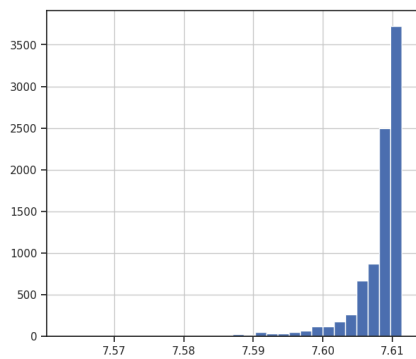
Исходное распределение для признака числового признака enginePower :

```
diagnostic_plots(data, 'release_year')
```



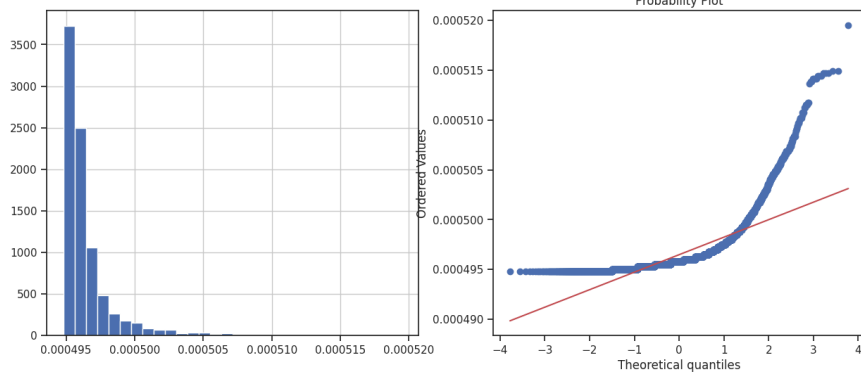
✓ Логарифмическое преобразование

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



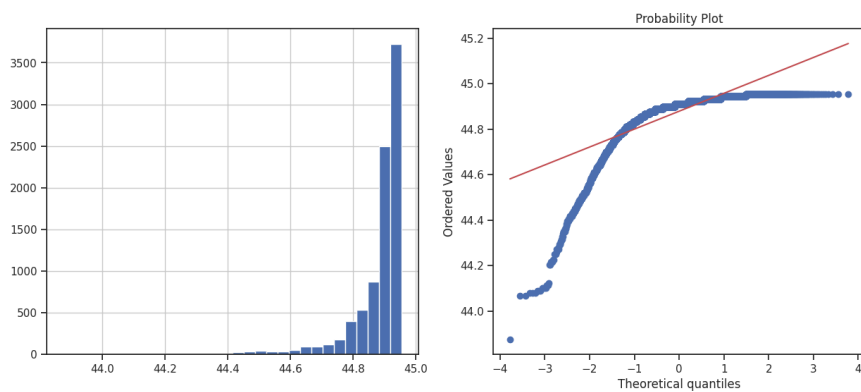
✓ Обратное преобразование

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

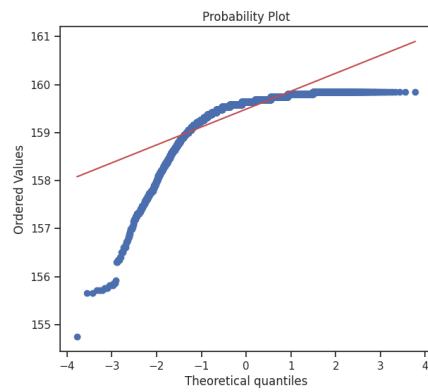
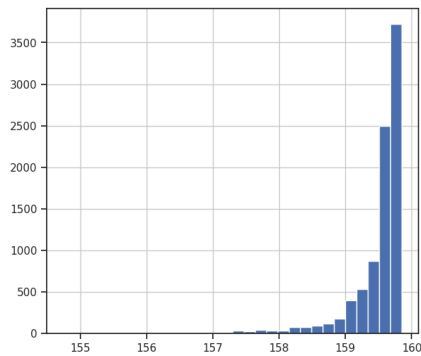
✓ Преобразование с использованием квадратного корня

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



✓ Возведение в степень

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

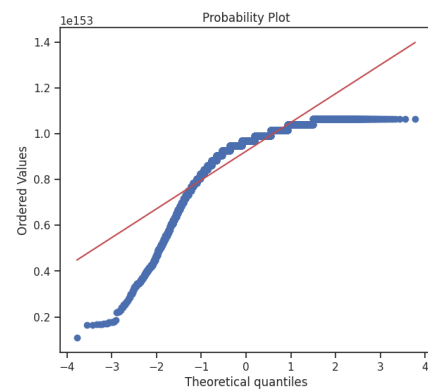
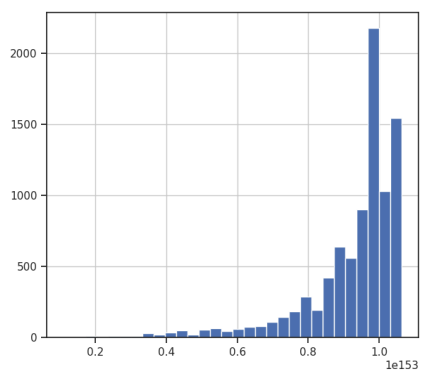


✓ Преобразование Бокса-Кокса

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



```
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:176: RuntimeWarning: c
x = um.multiply(x, x, out=x)
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:187: RuntimeWarning: c
ret = umr_sum(x, axis, dtype, out, keepdims=keepdims, where=where)
Оптимальное значение  $\lambda$  = 46.79897643237659
```



✓ Преобразование Йео-Джонсона

