# Лабораторная работа №2: "Обработка признаков (часть 1)."

## ИУ5-23 Зорин Арсений

**Задание:**

- Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных;
- Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
  - устранение пропусков в данных;
  - кодирование категориальных признаков;
  - нормализацию числовых признаков.

In [435…

```python
import numpy as np
import pandas as pd
import scipy.stats as stats
from category_encoders.count import CountEncoder as ce_CountEncoder
from sklearn.preprocessing import LabelEncoder

import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
plt.rcParams['figure.dpi'] = 300
import seaborn as sns
sns.set_palette('husl')

data = pd.read_csv('../datasets/spotify/RU.csv').explode('artists')
```

In [436…

```python
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()
```

In [437…

```python
data.head()
```

Out[437…

| | id | name | popularity | duration_ms | explicit | artists | id_artists | release_date |
|---|---|---|---|---|---|---|---|---|
| **0** | 35iwgR4jXetI318WEWsa1Q | Carve | 6 | 126903 | 0 | ['Uli'] | ['45tIt06XoI0Iio4LBEVpls'] | 1922-02-22 |
| **1** | 021ht4sdgPcrDgSk7JTbKY | Capítulo 2.16 - Banquero Anarquista | 0 | 98200 | 0 | ['Fernando Pessoa'] | ['14jtPCOoNZwquk5wd9DxrY'] | 1922-06-01 |
| **2** | 07A5yehtSnoedViJAZkNnc | Vivo para Quererte - Remasterizado | 0 | 181640 | 0 | ['Ignacio Corsini'] | ['5LiOoJbxVSAMkBS2fUm3X2'] | 1922-03-21 |
| **3** | 08FmqUhxtyLTn6pAh6bk45 | El Prisionero - Remasterizado | 0 | 176907 | 0 | ['Ignacio Corsini'] | ['5LiOoJbxVSAMkBS2fUm3X2'] | 1922-03-21 |
| **4** | 08y9GfoqCWfOGsKdwojr5e | Lady of the Evening | 0 | 163080 | 0 | ['Dick Haymes'] | ['3BiJGZsyX9sJchTqcSA7Su'] | 1922 |

In [438…

```python
data['release_year'] = pd.to_datetime(data.release_date, format='%Y-%m-%d', errors='ignore').dt.year
data['artists'] = data['artists'].apply(eval)
data = data.explode('artists')
```

In [439…

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

Out[439…

```
id                  0
name                1
popularity          0
duration_ms         0
explicit            0
artists             0
id_artists          0
release_date        0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
release_year        0
dtype: int64
```

В используемых данных отстутствуют пропуски. Однако в образовательных целях удалим 10% значений из столбца с валентностью (valence).

In [440…

```python
mask = np.random.choice([True, False], size=data['valence'].size, p=[0.1,0.9])
modified = data.copy()
modified['valence'] = modified['valence'].mask(mask)
```

In [441…

```python
modified.isnull().sum()
```

Out[441…

```
id                      0
name                    1
popularity              0
duration_ms             0
explicit                0
artists                 0
id_artists              0
release_date            0
danceability            0
energy                  0
key                     0
loudness                0
mode                    0
speechiness             0
acousticness            0
instrumentalness        0
liveness                0
valence             12475
tempo                   0
time_signature          0
release_year            0
dtype: int64
```

In [442…

```python
modified['valence'] = modified['valence'].fillna(modified['valence'].median())
modified.isnull().sum()
```

Out[442…

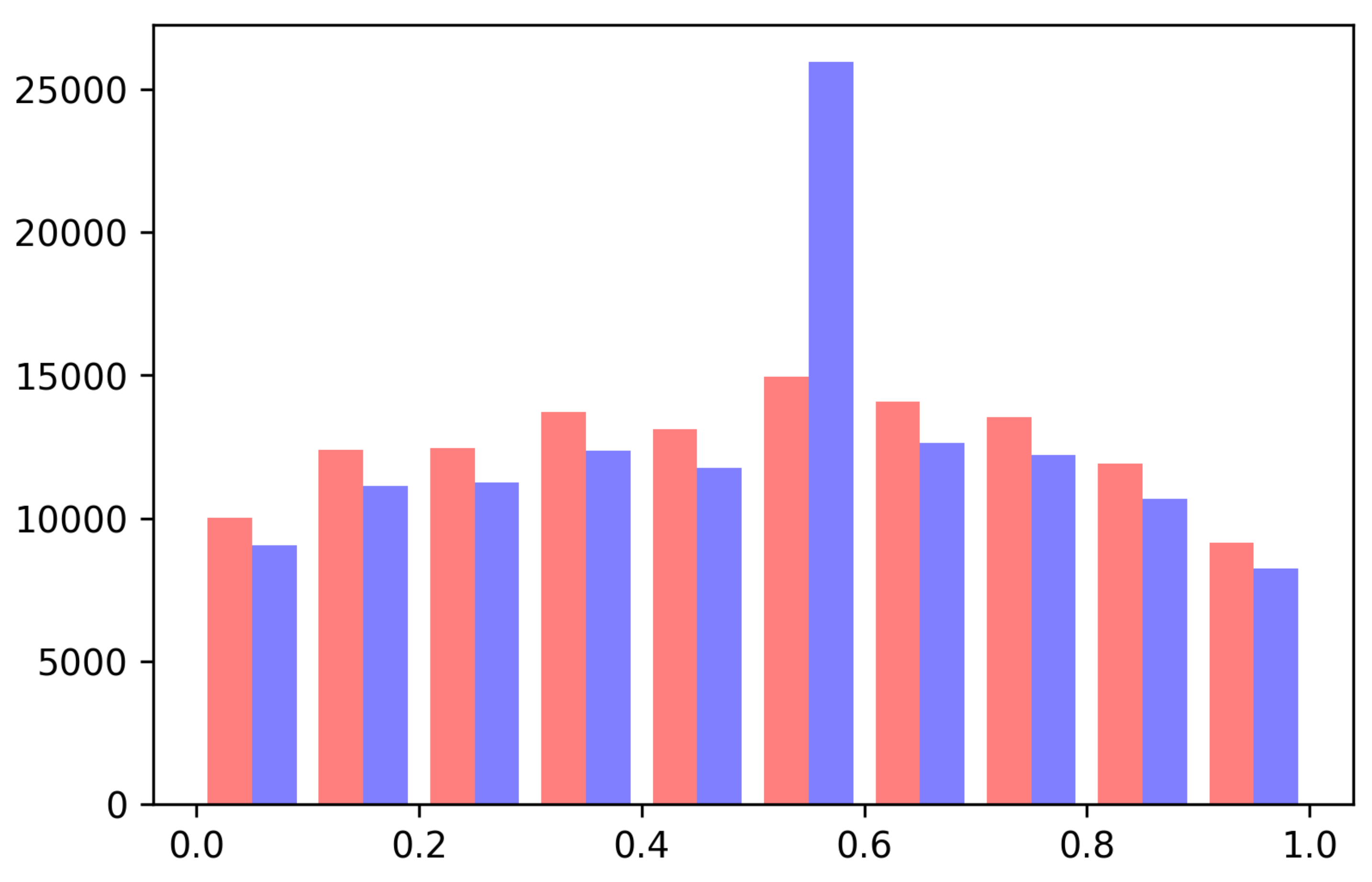
```
id                  0
name                1
popularity          0
duration_ms         0
explicit            0
artists             0
id_artists          0
release_date        0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
release_year        0
dtype: int64
```

In [443…

```python
# fig, ax = plt.subplots()
# sns.displot(data, x="valence", label="real", kind="kde", ax=ax)
# sns.displot(modified, x="valence", label="filled_with_median", kind="kde", ax=ax)
plt.hist([data['valence'], modified['valence']], color=['r','b'], alpha=0.5)
```

Out[443…

```
(array([[10003., 12393., 12469., 13720., 13119., 14947., 14074., 13542.,
         11919.,  9131.],
        [ 9064., 11144., 11241., 12353., 11762., 25969., 12647., 12228.,
         10676.,  8233.]]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <a list of 2 BarContainer objects>)
```



Нормализуем параметры композиции

Удалим строки содержащие 0.

In [444…

```python
data[data.isin([0])].count() / data.size * 100
```

Out[444…

```
id                  0.000000
name                0.000000
popularity          1.257307
duration_ms         0.000000
explicit            4.516052
artists             0.000000
id_artists          0.000000
release_date        0.000000
danceability        0.004218
energy              0.000646
key                 0.577203
loudness            0.000000
mode                1.608226
speechiness         0.004218
acousticness        0.001102
instrumentalness    1.051847
liveness            0.000912
valence             0.005130
tempo               0.004218
time_signature      0.004218
release_year        0.000000
dtype: float64
```

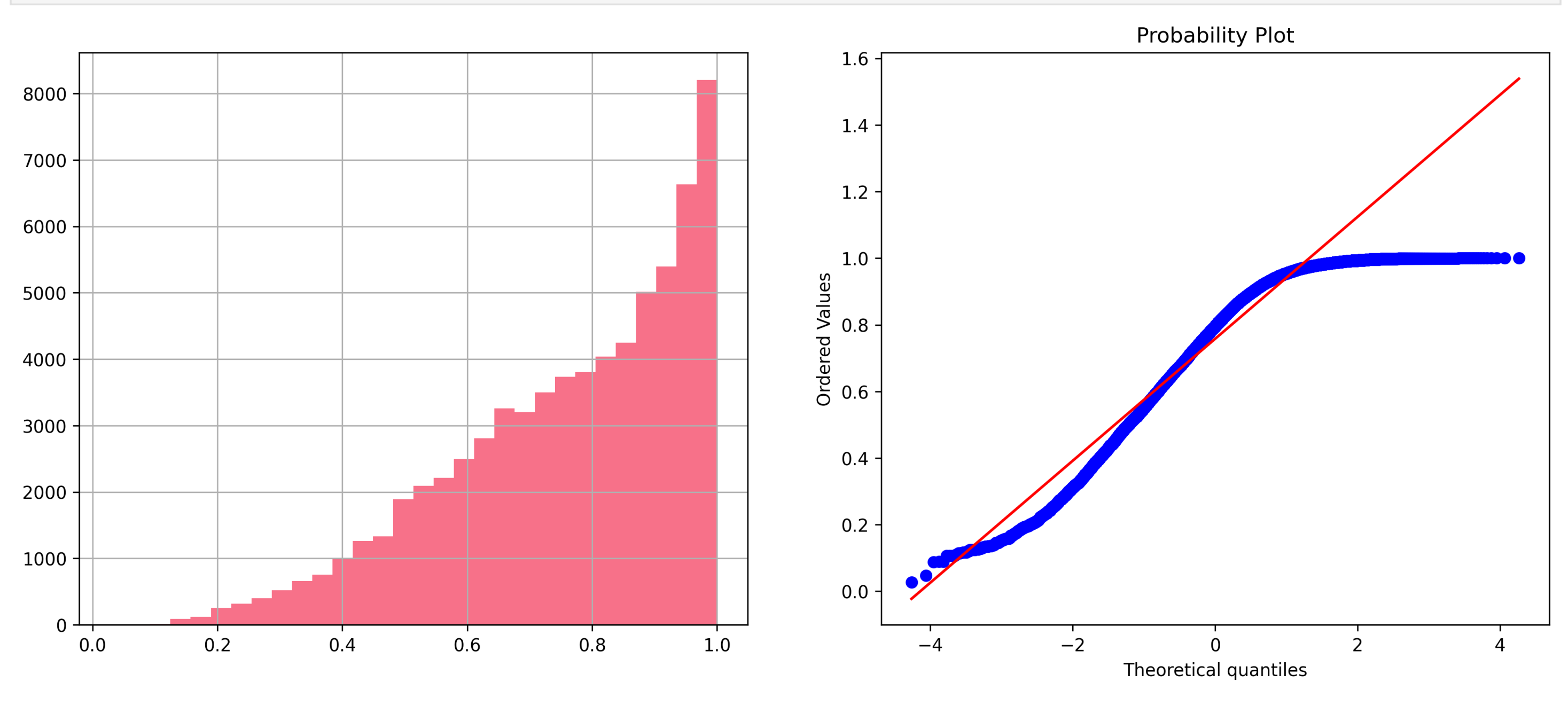
In [445…

```python
num_cols = ['acousticness', 'danceability', 'energy', 'duration_ms', 'instrumentalness', 'valence',
            'popularity', 'tempo', 'liveness', 'loudness', 'speechiness']
data = data[(data[num_cols] != 0).all(axis=1)]
```

Нормализуем числовые признаки.

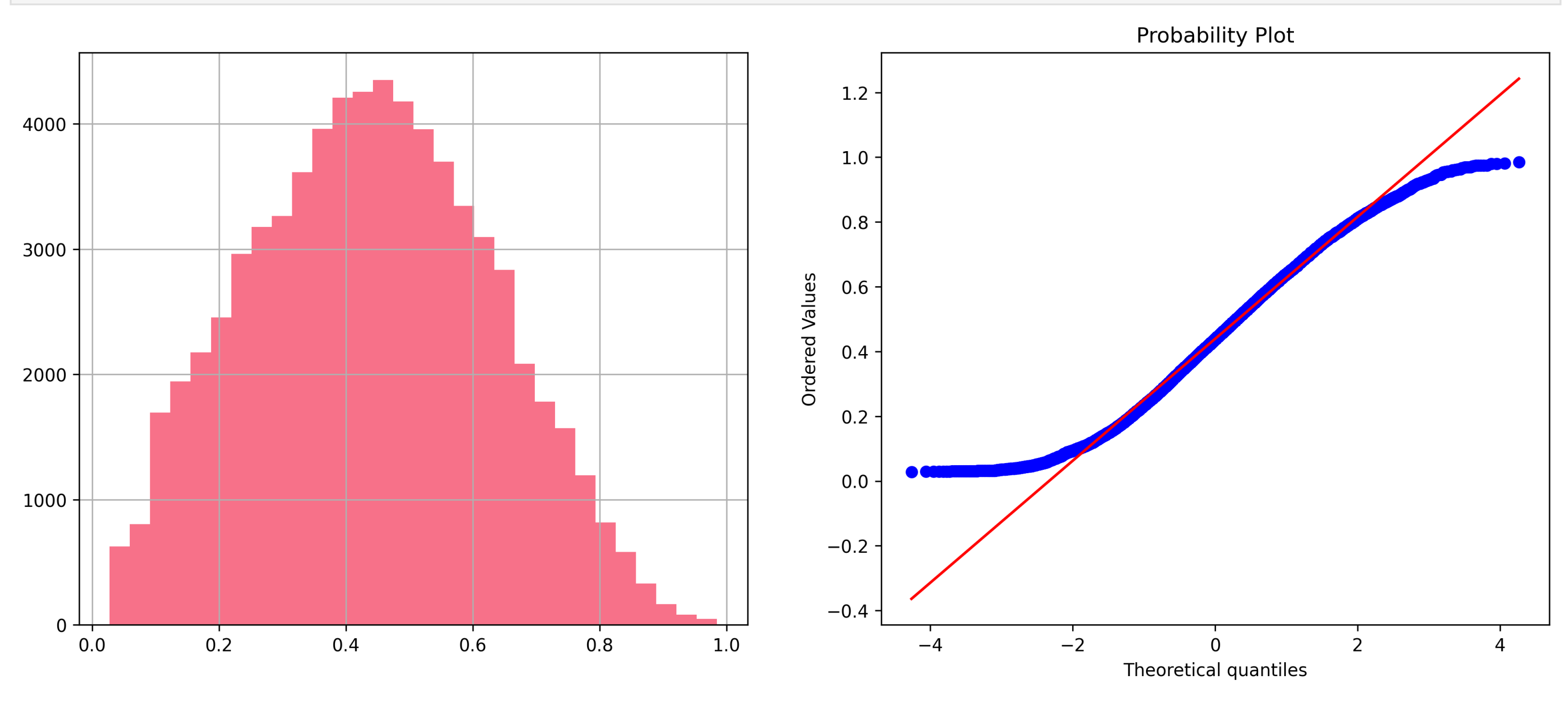
In [446…

```python
data['energy']= data['energy'] ** 0.333
diagnostic_plots(data, 'energy')
```



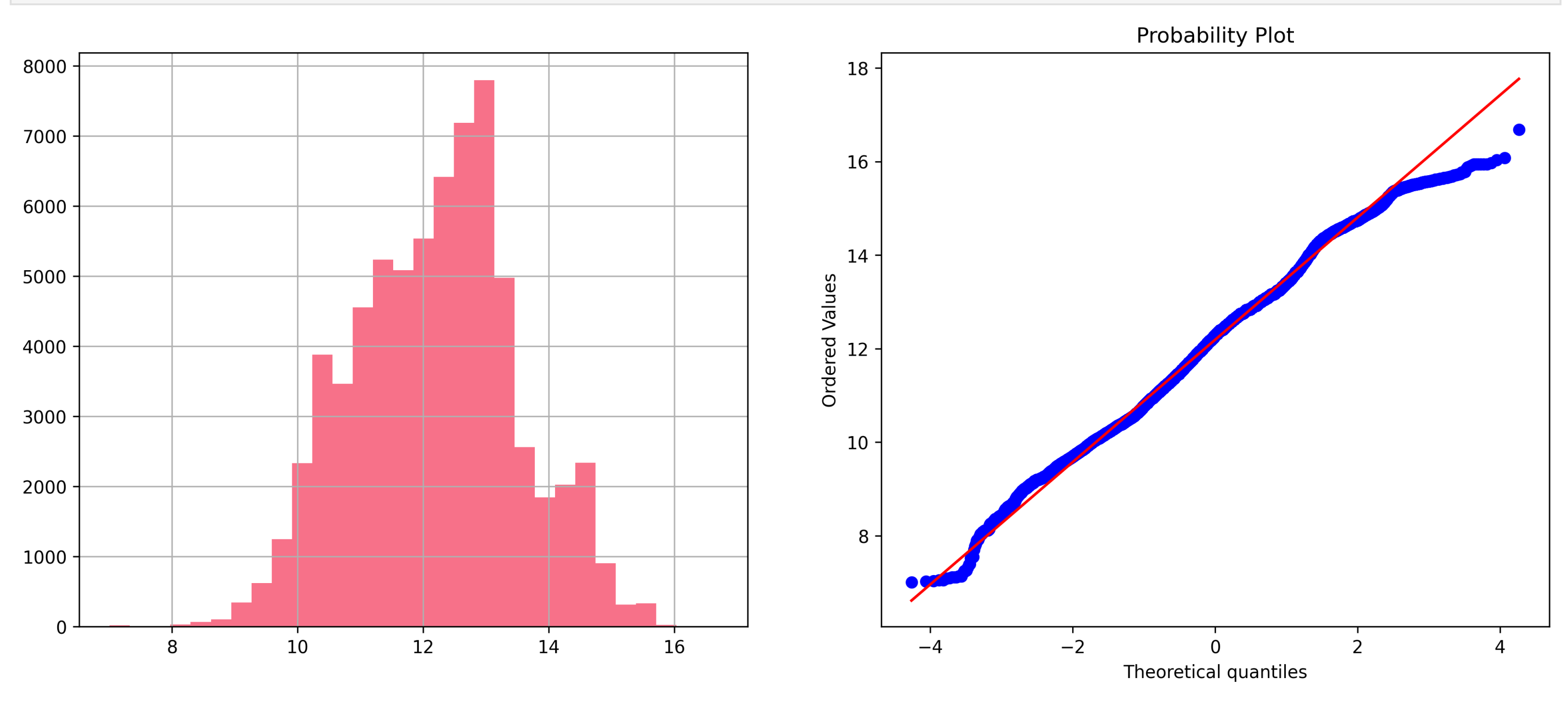
In [447…

```python
data['danceability'] = data['danceability'] ** 1.25
diagnostic_plots(data, 'danceability')
```



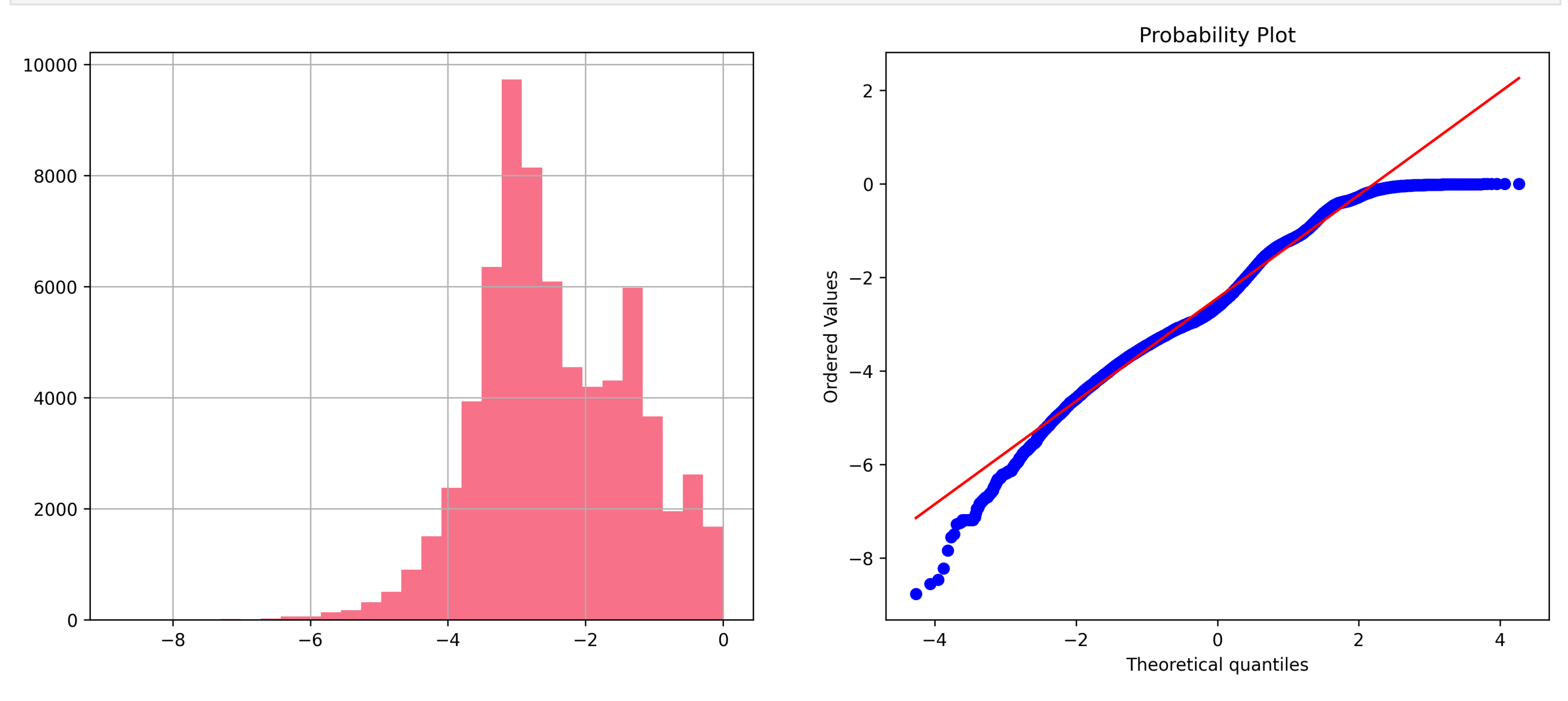
In [448…

```python
data['tempo'], _ = stats.boxcox(data['tempo'])
diagnostic_plots(data, 'tempo')
```



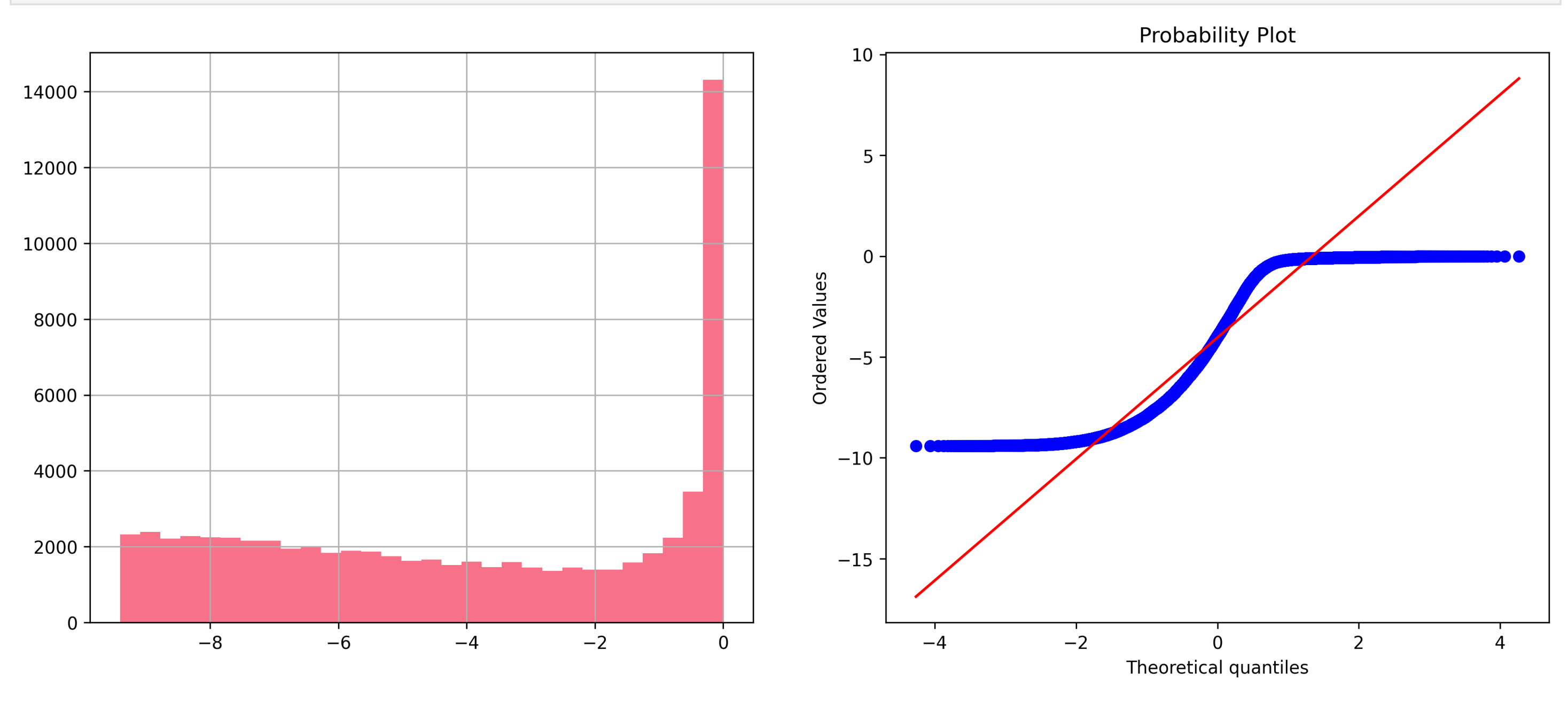
In [449…

```python
data['liveness'], _ = stats.boxcox(data['liveness'])
diagnostic_plots(data, 'liveness')
```



In [450…

```python
data['instrumentalness'], _ = stats.boxcox(data['instrumentalness'])
diagnostic_plots(data, 'instrumentalness')
```

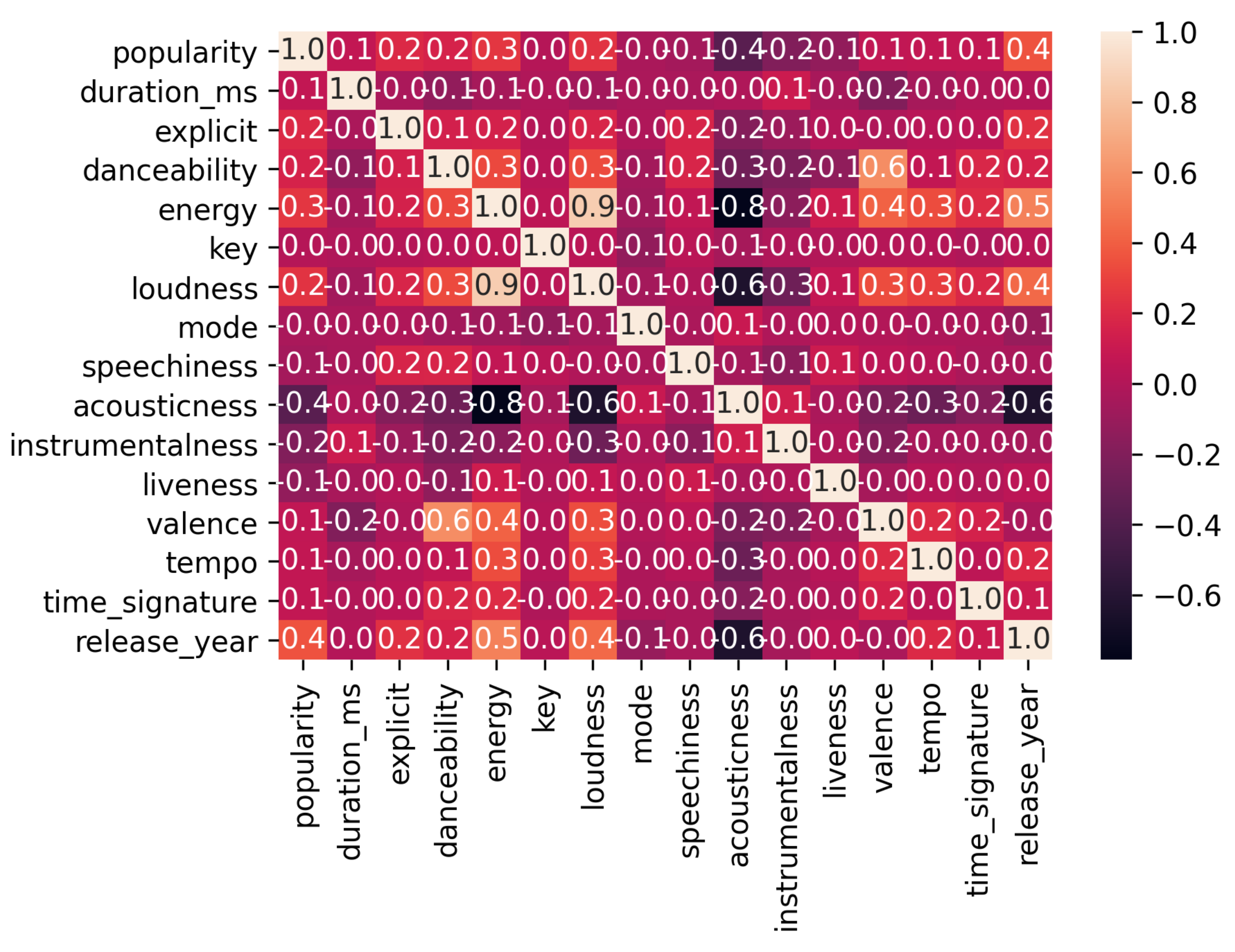


In [451…

```python
sns.heatmap(data.corr(), annot=True, fmt='.1f')
```

Out[451…

```
<AxesSubplot:>
```



In [452…

```python
data = data.drop(['duration_ms', 'mode', 'speechiness', 'valence', 'tempo',
                  'time_signature', 'key', 'liveness', 'loudness', 'acousticness'], axis=1)
```

Кодирование категориальных признаков.

In [453…

```python
data['trend'] = data[['energy', 'danceability', 'instrumentalness']].idxmax(axis=1)
data['trend'] = LabelEncoder().fit_transform(data['trend'])
data.head()
```

Out[453…

| | id | name | popularity | explicit | artists | id_artists | release_date | danceability |
|---|---|---|---|---|---|---|---|---|
| **0** | 35iwgR4jXetI318WEWsa1Q | Carve | 6 | 0 | Uli | ['45tIt06XoI0Iio4LBEVpls'] | 1922-02-22 | 0.578029 |
| **13** | 0QiT0Oo5QdLXdFw6RDOj7h | Tu Verras Montmartre | 1 | 0 | Lucien Boyer | ['4mSouLpNSEY1d7OdljlFiP'] | 1922 | 0.643716 |
| **26** | 112adU33vo4C1eRZct2hWy | Nuits De Chine | 4 | 0 | Louis Lynel | ['28pbliOohRRZjqpAM9iqYM'] | 1922 | 0.338113 |
| **149** | 2wAfHM7Whz67VFbdanhZlk | Nobody Knows You When You're Down and Out | 41 | 0 | Bessie Smith | ['5ESobCkc6JI4tIMxQttqeg'] | 1923 | 0.543514 |
| **150** | 7lRFR5GJCxK87ZbVMtQSeS | Ain't Misbehavin' | 28 | 0 | Louis Armstrong | ['19eLuQmk9aCobbVDHc6eek'] | 1923 | 0.600519 |