**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 и системы управления

КАФЕДРА Системы обработки информации и управления

Отчет по лабораторной работе № 1

**«Разведочный анализ данных. Исследование и визуализация данных.»**
по дисциплине «Технологии машинного обучения»

Студент ИУ5-61Б _____ Е.И. Бирюкова
(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель _____ А.Н. Нардид
(Подпись, дата) (И.О.Фамилия)

**Москва**

**2025**

**Цель работы**

Изучение различных методов визуализация данных. Построение основных графиков, входящих в этап разведочного анализа данных.

**Задание**

1. Выбрать набор данных (датасет). Список свободно распространяемых датасетов можно найти на сайте.
2. Для первой лабораторной работы рекомендуется использовать датасет без пропусков в данных, например, из Scikit-learn.
3. Пример преобразования датасетов Scikit-learn в Pandas Dataframe можно посмотреть в примерах.
4. Для лабораторных работ не рекомендуется выбирать датасеты большого размера.
5. Создать ноутбук, который содержит следующие разделы:
   5.1. Текстовое описание выбранного Вами набора данных.
   5.2. Основные характеристики датасета.
   5.3. Визуальное исследование датасета.
   5.4. Информация о корреляции признаков.
6. Сформировать отчет и разместить его в своем репозитории на github.
   Отчет по лабораторной работе должен содержать:
   6.1. Титульный лист;
   6.2. Описание задания;
   6.3. Текст программы;
   6.4. Экранные формы с примерами выполнения программы.

**Ход лабораторной работы**

***1. Текстовое описание выбранного набора данных.***

Текстовое описание набора данных.

## 1) Текстовое описание набора данных

### Введение

В качестве набора данных мы будем использовать набор данных датасет "300 world tracks", который содержит информацию о 326 треках от 66 исполнителей в период с 1958 по 2019 год. Ссылка на датасет: https://www.kaggle.com/datasets/thebumpkin/300-world-music-tracks-with-spotify-data

Использованы аудио-характеристики Spotify для анализа эволюции музыкальных параметров, сравнения различных культур и определения взаимосвязей между популярностью и музыкальными чертами. Исследование включает в себя анализ распределений и корреляций данных.

Полученные результаты могут быть полезны для музыкантов, исследователей музыкальной культуры и специалистов в области анализа данных.

Датасет хранится в файле WorldHits.csv. Файл содержит следующие колонки:

```
Track - Название музыкального трека. (текст)
Artist - Имя исполнителя или группы. (текст)
Album - Название альбома. (текст)
Year - Год выпуска трека. (целое число)
Duration - Длина трека в миллисекундах. (целое число)
Time_Signature - Размер такта (beats per measure). (целое число)
Danceability - Показатель танцевальности (от 0.0 до 1.0). (вещественное число)
Energy - Показатель энергичности (от 0.0 до 1.0). (вещественное число)
Key - Музыкальная тональность трека. (целое число)
Loudness - Громкость трека в децибелах (dB). (вещественное число)
Mode - Мажорный (1) или минорный (0) лад. (целое число)
Speechiness - Наличие речи в треке (от 0.0 до 1.0). (вещественное число)
Acousticness - Показатель акустичности (от 0.0 до 1.0). (вещественное число)
Instrumentalness - Показатель наличия вокала (от 0.0 до 1.0). (вещественное число)
Liveness - Показатель живого звучания (от 0.0 до 1.0). (вещественное число)
Valence - Показатель музыкальной позитивности (от 0.0 до 1.0). (вещественное число)
Tempo - Темп трека в ударах в минуту (BPM). (вещественное число)
Popularity - Популярность трека на Spotify (от 0 до 100). (целое число)
```

Импортируем библиотеки с помощью import.

### Импорт библиотек

Импортируем библиотеки с помощью команды import.

[1]:
```
%matplotlib notebook
```

[*]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import statsmodels.stats.multitest
```

[*]:
```
from scipy import stats
```

Загрузим и очистим данные. Проверим на пропуски.

### Загрузка и очистка данных

Загрузим файлы датасета в помощью библиотеки Pandas. Проверим данные на предмет присутствия пропусков, выбросов и пр. Также немного преобразуем данные для более эффективного хранения памяти и удобного использования в будущем.

[4]:
```
df = pd.read_csv('WorldHits.csv')
```

[5]:
```
# Проверка на пропуски
print("Пропуски до обработки:")
print(df.isna().sum())
```

```
Пропуски до обработки:
Track               0
Artist              0
Album               0
Year                0
Duration            0
Time_Signature      0
Danceability        0
Energy              0
Key                 0
Loudness            0
Mode                0
Speechiness         0
Acousticness        0
Instrumentalness    0
Liveness            0
Valence             0
Tempo               0
```

Преобразуем числовую колонку Mode в категориальный признак.

[6]:
```
# Преобразование данных
mode_mapping = {0: 'minor', 1: 'major'}
df['Mode'] = df['Mode'].replace(mode_mapping)
df['Mode'] = df['Mode'].astype('category')
df.head(10)
```

[6]:

| | Track | Artist | Album | Year | Duration | Time_Signature | Danceability | Energy | Key | Loudness | Mode | Speechiness | Acousticness | Instrumentalness | Liver |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Release | Afro Celt Sound System | Volume 2: Release (Real World Gold) | 2005 | 456160 | 4 | 0.633 | 0.828 | 5 | -7.266 | minor | 0.0480 | 0.0216 | 0.055800 | 0.1 |
| **1** | Saor / Free / News from Nowhere | Afro Celt Sound System | Vol. 1: Sound Magic (Real World Gold) | 1999 | 501093 | 4 | 0.511 | 0.524 | 7 | -10.504 | major | 0.0305 | 0.0260 | 0.879000 | 0.1 |
| **2** | When You're Falling | Afro Celt Sound System | Volume 3 : Further In Time (Real World Gold) | 2003 | 314160 | 4 | 0.638 | 0.822 | 11 | -7.305 | major | 0.0380 | 0.0508 | 0.000025 | 0.0 |

## 2. *Основные характеристики датасета.*

## 2) Основные характеристики датасета.

Посчитаем основные статистики

[19]:
```
# Первые 5 строк датасета
df.head()
```

[19]:

| | Track | Artist | Album | Year | Duration | Time_Signature | Danceability | Energy | Key | Loudness | Mode | Speechiness | Acousticness | Instrumentalness | Livenes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Release | Afro Celt Sound System | Volume 2: Release (Real World Gold) | 2005 | 456160 | 4 | 0.633 | 0.828 | 5 | -7.266 | minor | 0.0480 | 0.0216 | 0.055800 | 0.108 |
| **1** | Saor / Free / News from Nowhere | Afro Celt Sound System | Vol. 1: Sound Magic (Real World Gold) | 1999 | 501093 | 4 | 0.511 | 0.524 | 7 | -10.504 | major | 0.0305 | 0.0260 | 0.879000 | 0.106 |
| **2** | When You're Falling | Afro Celt Sound System | Volume 3 : Further In Time (Real World Gold) | 2003 | 314160 | 4 | 0.638 | 0.822 | 11 | -7.305 | major | 0.0380 | 0.0508 | 0.000025 | 0.089 |
| **3** | Whirl-Y- | Afro Celt | Vol. 1: Sound Magic (Real | 1999 | 441200 | 4 | 0.645 | 0.810 | 2 | -8.133 | major | 0.0381 | 0.1320 | 0.395000 | 0.082 |

Размер датасета и список колонок с типами данных. Статистики для категориального признака.

[20]:
```
# Размер датасета - 326 строк, 18 колонок
df.shape
```

[20]:
```
(326, 18)
```

[21]:
```
total_count = df.shape[0]
print('Всего строк: {}'.format(total_count))
# Список колонок
# df.columns
# Список колонок с типами данных
df.dtypes
```

```
Всего строк: 326
```

[21]:
```
Track                object
Artist               object
Album                object
Year                  int64
Duration              int64
Time_Signature        int64
Danceability        float64
Energy              float64
Key                   int64
Loudness            float64
Mode               category
Speechiness         float64
Acousticness        float64
Instrumentalness    float64
Liveness            float64
Valence             float64
Tempo               float64
Popularity            int64
dtype: object
```

[22]:
```
# Основные статистики для категориальных признаков
print(df.select_dtypes('category').describe())
```

```
        Mode
count    326
unique     2
top    major
freq     206
```

Основные статистики для числовых признаков и уникальные значения для целевого признака.

[23]:
```
# Основные статистики для числовых признаков
print(df.describe())
```

```
              Year      Duration  Time_Signature  Danceability      Energy  \
count   326.000000  3.260000e+02      326.000000    326.000000  326.000000
mean   1992.407975  3.337062e+05        3.889571      0.522381    0.502092
std      12.680298  2.680486e+05        0.415191      0.170469    0.259283
min    1958.000000  4.188000e+04        1.000000      0.078800    0.005620
25%    1985.000000  2.124700e+05        4.000000      0.404250    0.292250
50%    1995.000000  2.648330e+05        4.000000      0.542000    0.489500
75%    2002.000000  3.498832e+05        4.000000      0.657750    0.729000
max    2019.000000  3.060650e+06        5.000000      0.946000    0.985000

              Key    Loudness  Speechiness  Acousticness  Instrumentalness  \
count  326.000000  326.000000   326.000000    326.000000        326.000000
mean     5.438650  -11.799755     0.061305      0.519446          0.240264
std      3.580405    5.159427     0.065734      0.331782          0.344821
min      0.000000  -36.178000     0.023800      0.000002          0.000000
25%      2.000000  -14.645000     0.034225      0.196500          0.000031
50%      5.000000  -11.055500     0.041600      0.558000          0.014250
75%      9.000000   -7.894500     0.059900      0.830750          0.512000
max     11.000000   -1.656000     0.698000      0.991000          0.967000

         Liveness     Valence       Tempo  Popularity
count  326.000000  326.000000  326.000000  326.000000
mean     0.215337    0.503624  115.968460   28.346626
std      0.210846    0.270387   28.174109   15.960512
min      0.028200    0.030900   44.757000    0.000000
25%      0.093600    0.283250   94.757500   17.000000
50%      0.124000    0.516000  112.277000   28.000000
75%      0.257500    0.732250  131.378250   38.000000
max      0.987000    0.964000  200.076000   81.000000
```

[24]:
```
# Определим уникальные значения для целевого признака
df['Mode'].unique()
```

[24]:
```
['minor', 'major']
Categories (2, object): ['major', 'minor']
```

Целевой признак является категориальным и содержит только значения "minor" и "major".

## 3. Визуальное исследование датасета.

Для визуального исследования могут быть использованы различные виды диаграмм, мы построим только некоторые варианты диаграмм, которые используются достаточно часто.

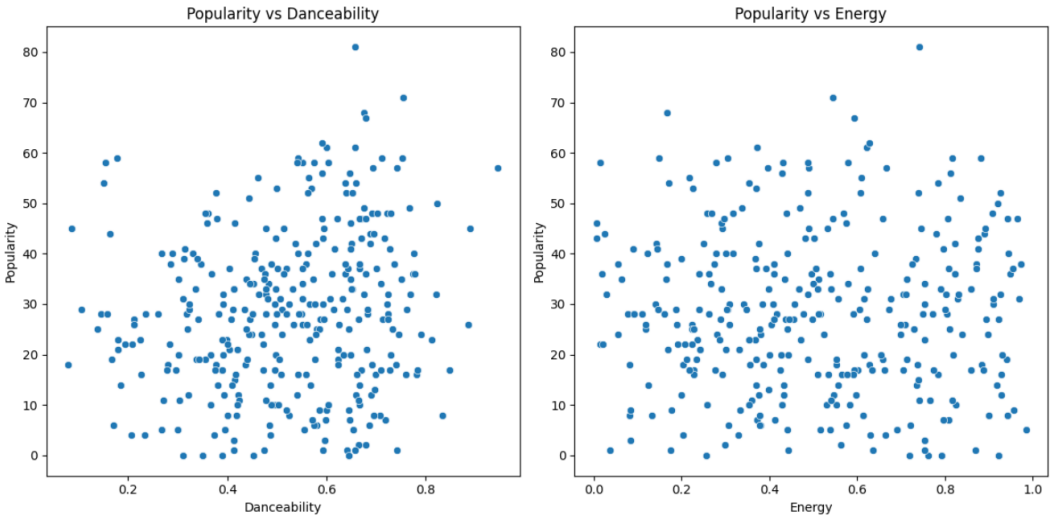
### 3.1. Графики рассеяния и анализ корреляции

**Графики рассеяния**

Позволяет построить распределение двух колонок данных и визуально обнаружить наличие зависимости.

Построим графики рассеяния, показывающие, есть ли связь между популярностью и танцевальностью/энергичностью треков. Построим матрицу корреляций, которая количественно оценивает связь между этими признаками.

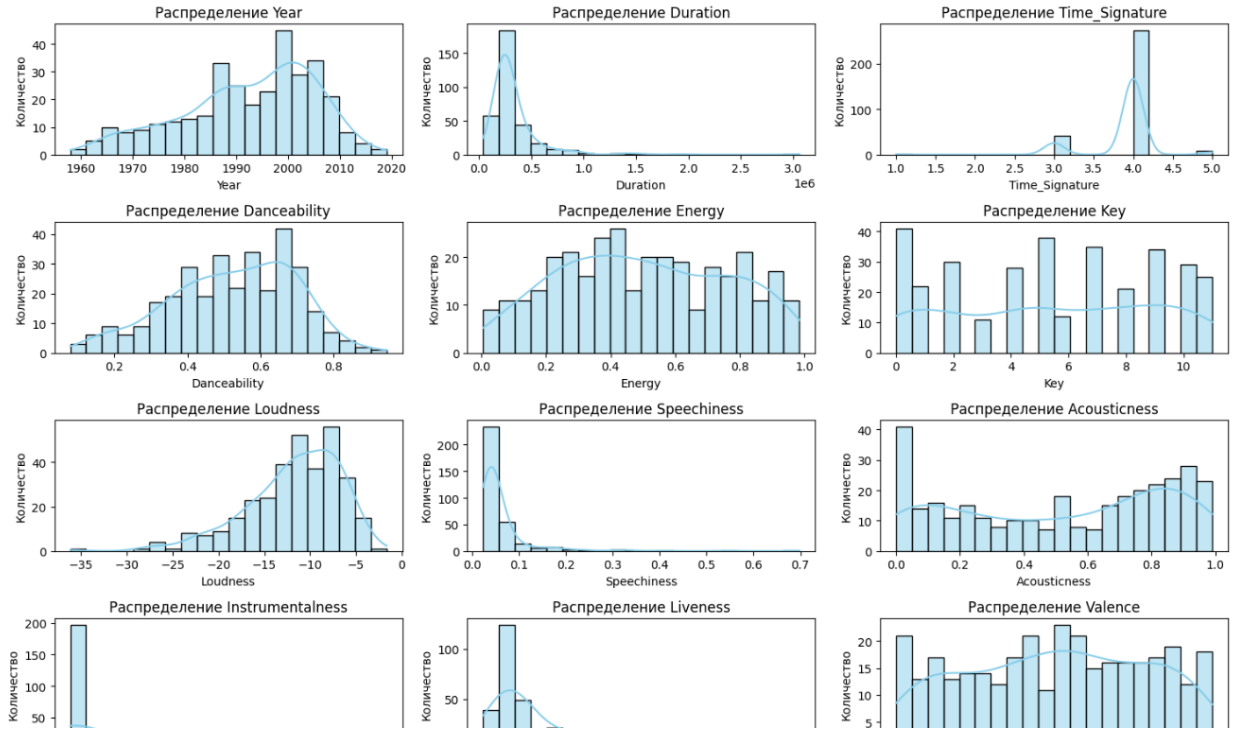
```
%matplotlib inline

# Визуализация
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.scatterplot(x='Danceability', y='Popularity', data=df)
plt.title('Popularity vs Danceability')
plt.subplot(1, 2, 2)
sns.scatterplot(x='Energy', y='Popularity', data=df)
plt.title('Popularity vs Energy')
plt.tight_layout()
plt.show()

# Анализ корреляции
corr_dance_energy_pop = df[['Popularity', 'Danceability', 'Energy']].corr(numeric_only=True)
print("Корреляция между популярностью и танцевальностью/энергичностью:\n", corr_dance_energy_pop['Popularity'].drop('Popularity'))
```



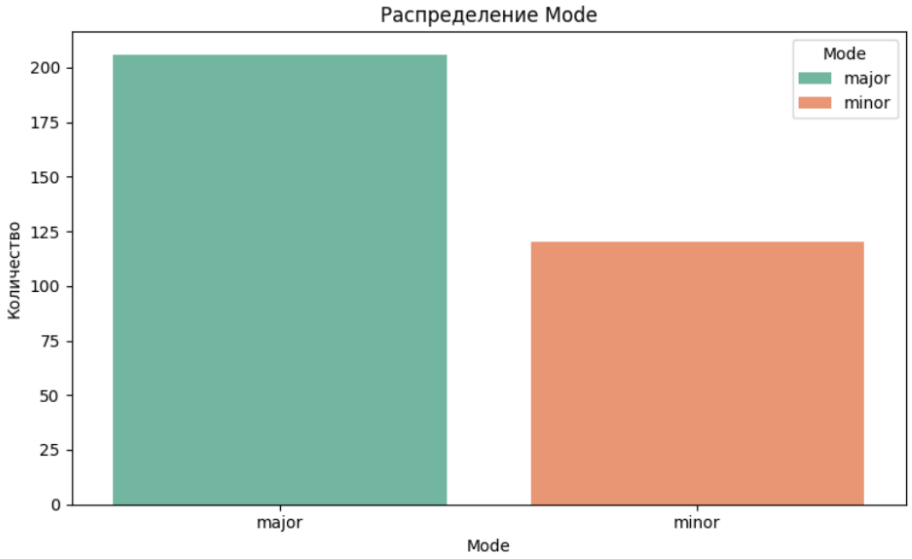

```
Корреляция между популярностью и танцевальностью/энергичностью:
 Danceability    0.158759
Energy          -0.010802
Name: Popularity, dtype: float64
```

## 3.2. Гистограмма

```
# Распределение числовых признаков
num_cols = df.select_dtypes(include=['float64', 'int64']).columns
fig, axes = plt.subplots(len(num_cols) // 3 + (len(num_cols) % 3 > 0), 3, figsize=(15, 12))
axes = axes.flatten()

for ax, col in zip(axes, num_cols):
    sns.histplot(df[col], ax=ax, kde=True, bins=20, color='skyblue')
    ax.set_title(f'Распределение {col}')
    ax.set_ylabel('Количество')

plt.tight_layout()
plt.show()
```



```
# Распределение категориальных признаков
cat_cols = df.select_dtypes(include=['category']).columns
fig, axes = plt.subplots(len(cat_cols), 1, figsize=(8, 5 * len(cat_cols)))
if hasattr(axes, 'flatten'):
    axes = axes.flatten()
else:
   axes = [axes]

for ax, col in zip(axes, cat_cols):
    sns.countplot(x=df[col], ax=ax, palette='Set2', hue=df[col], legend=True)
    ax.set_title(f'Распределение {col}')
    ax.set_ylabel('Количество')

plt.tight_layout()
plt.show()
```

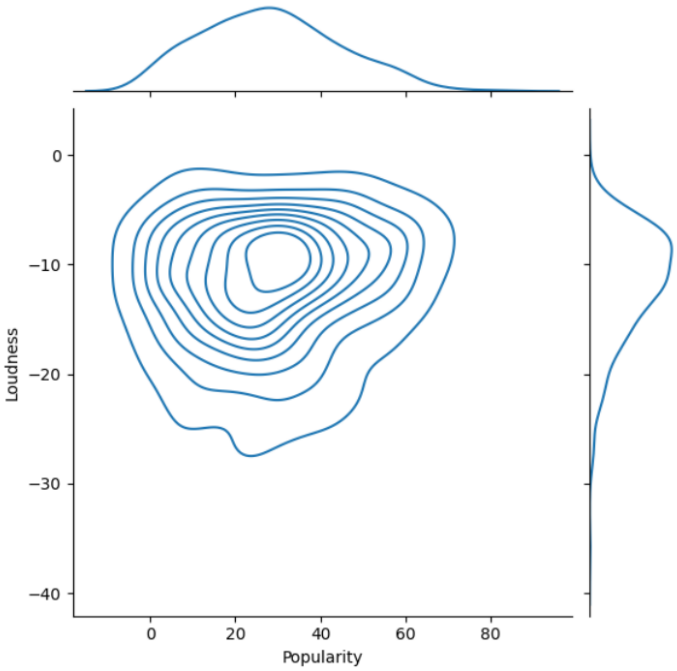



## 3.3. Jointplot

Комбинация гистограмм и диаграмм рассеивания. Диаграмма показывает совместное распределение популярности и громкости треков Spotify с помощью KDE, визуализируя плотность их комбинаций и одномерные распределения каждого признака. Анализ контуров и пиков позволяет оценить наличие и характер взаимосвязи между популярностью и громкостью треков.

```
sns.jointplot(x='Popularity', y='Loudness', data=df, kind = 'kde')
```

```
<seaborn.axisgrid.JointGrid at 0x1db5328e510>
```



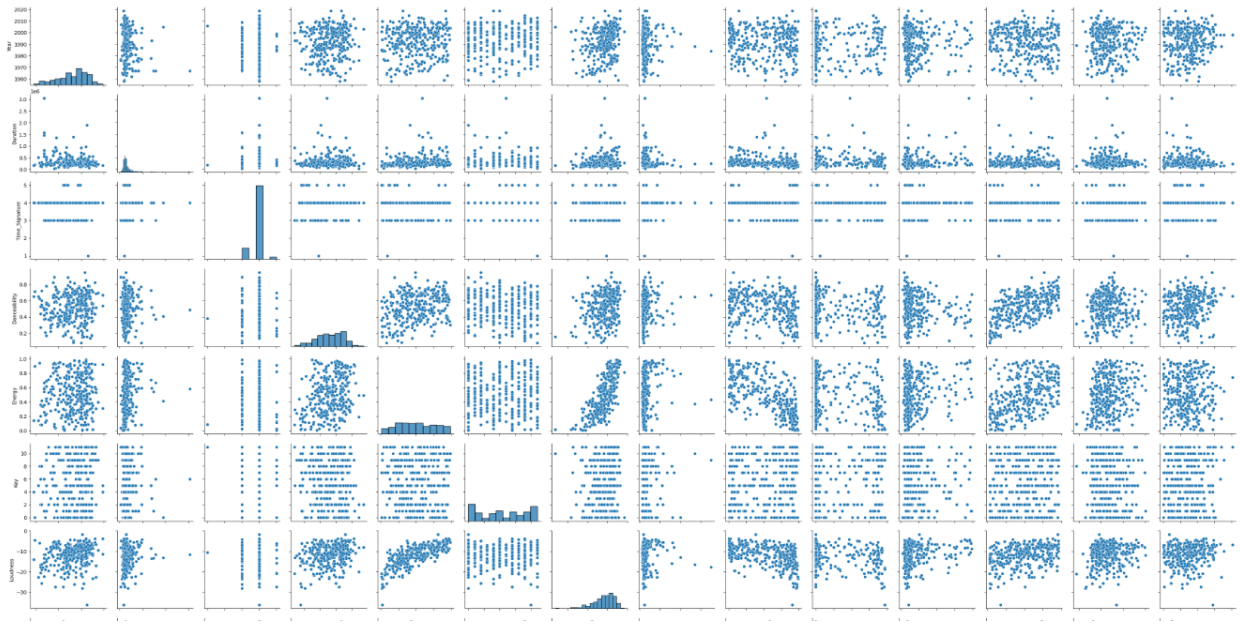
## 3.4. Парные диаграммы

Комбинация гистограмм и диаграмм рассеивания для всего набора данных.

Выводится матрица графиков. На пересечении строки и столбца, которые соответствуют двум показателям, строится диаграмма рассеивания. В главной диагонали матрицы строятся гистограммы распределения соответствующих показателей.

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x1db53349e80>
```



С помощью параметра "hue" возможна группировка по значениям какого-либо признака.

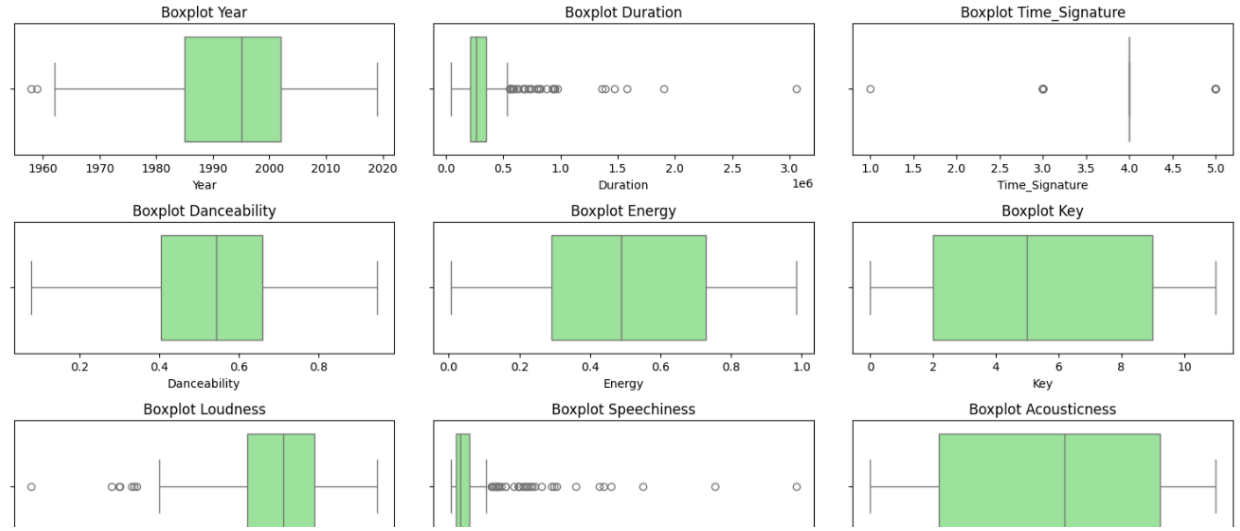
```
sns.pairplot(df, hue="Mode")
```

```
<seaborn.axisgrid.PairGrid at 0x1db62884f50>
```

## 3.5. Ящик с усами

Для непрерывных переменных можно использовать boxplot для идентификации выбросов. Ящик с усами (Boxplot) показывает распределение данных по квартилям, медиане, а также выявляет выбросы. Он позволяет быстро оценить центральную тенденцию, разброс и симметричность данных.

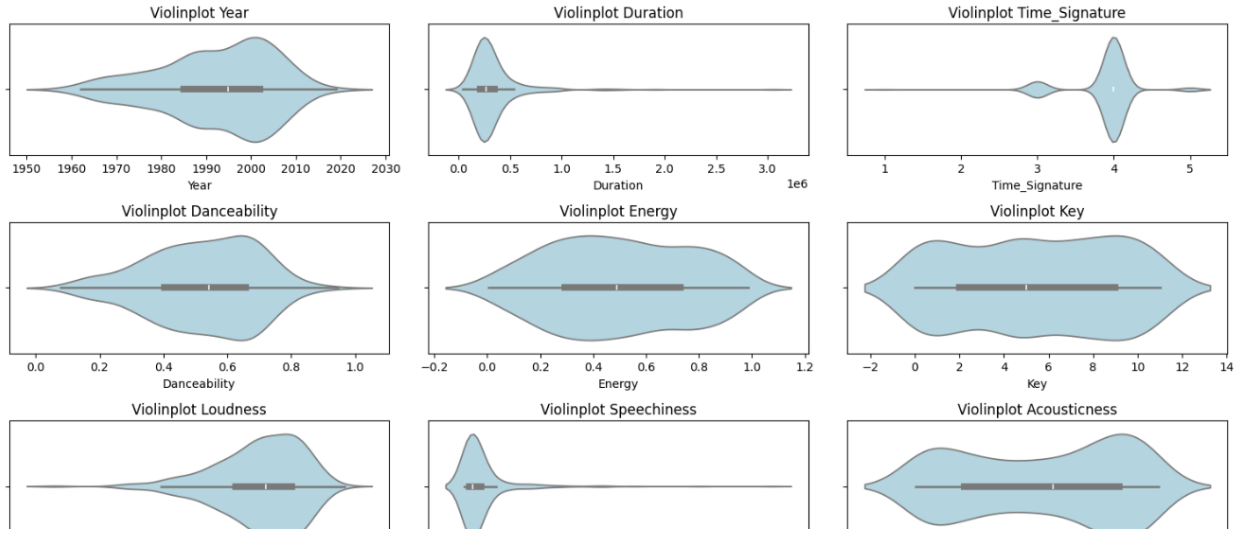
```
# Boxplot для числовых признаков
fig, axes = plt.subplots(len(num_cols) // 3 + (len(num_cols) % 3 > 0), 3, figsize=(15, 12))
axes = axes.flatten()

for ax, col in zip(axes, num_cols):
    sns.boxplot(x=df[col], ax=ax, color='lightgreen')
    ax.set_title(f'Boxplot {col}')

plt.tight_layout()
plt.show()
```



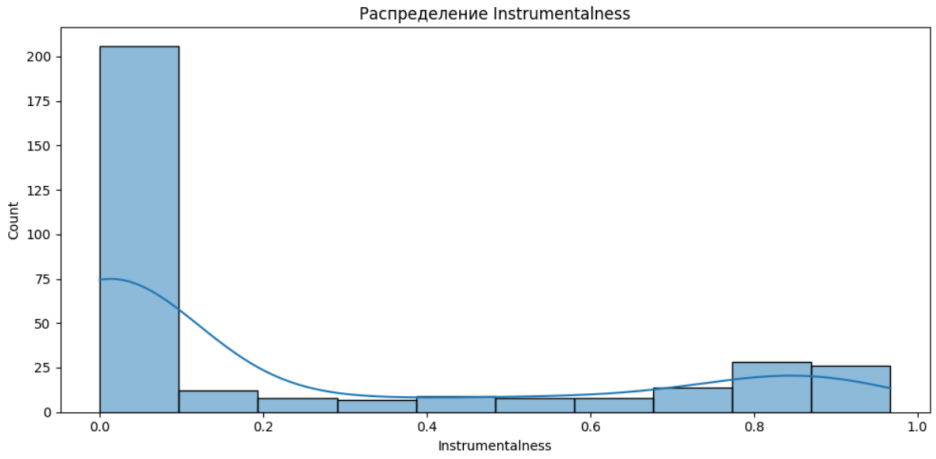

## 3.6. Violin plot

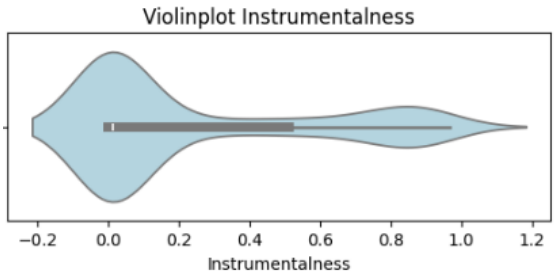
Для непрерывных переменных можно использовать violinplot для демонстрации плотности вероятности. Скрипичный график сочетает в себе ящик с усами и оценку плотности ядра (KDE), демонстрируя как квартили и выбросы, так и форму распределения данных. Он наглядно показывает концентрацию значений и позволяет сравнивать распределения разных групп.

```
# Violinplot для числовых признаков
fig, axes = plt.subplots(len(num_cols) // 3 + (len(num_cols) % 3 > 0), 3, figsize=(15, 12))
axes = axes.flatten()

for ax, col in zip(axes, num_cols):
    sns.violinplot(x=df[col], ax=ax, color='lightblue')
    ax.set_title(f'Violinplot {col}')

plt.tight_layout()
plt.show()
```



[27]:
```
# проверим скрипичную диаграмму для показателя наличия вокала
fig, ax = plt.subplots(figsize=(10, 5))
ax.set_title('Распределение Instrumentalness')
sns.histplot(df['Instrumentalness'], kde=True, ax=ax)
plt.tight_layout()
plt.show()
```




# 4. Информация о корреляции признаков.

## Проверим корреляцию признаков.

Проверка корреляции признаков позволяет решить две задачи:

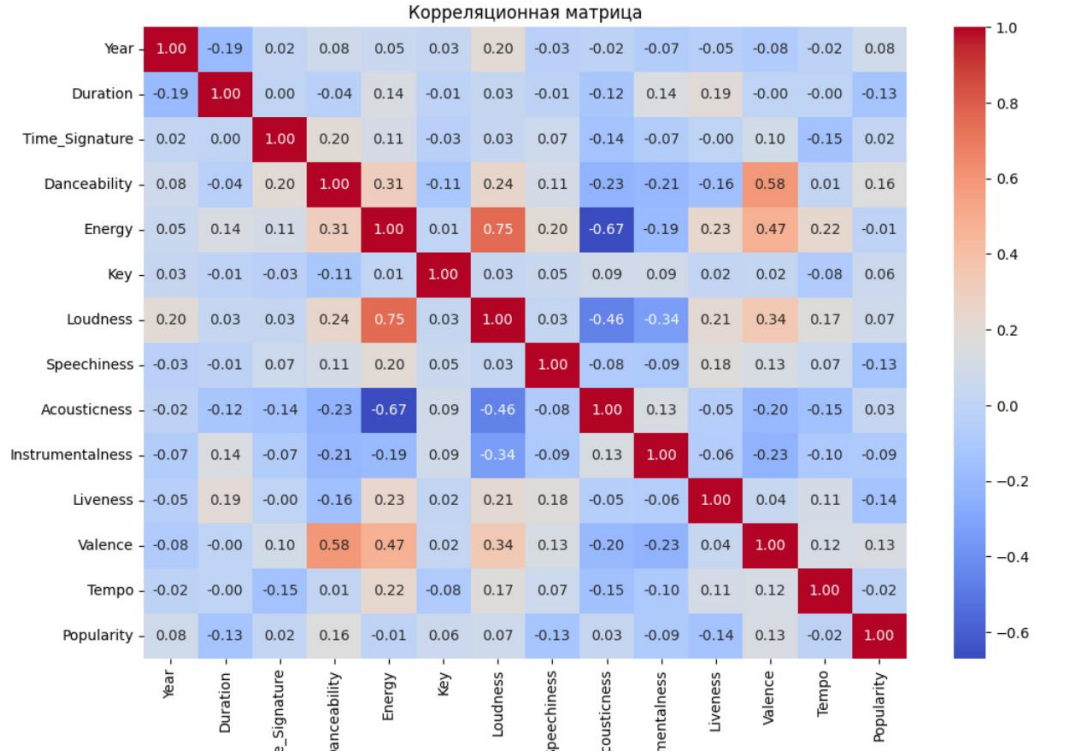
1. Понять какие признаки (колонки датасета) наиболее сильно коррелируют с целевым признаком (в нашем примере это колонка "Mode"). Именно эти признаки будут наиболее информативными для моделей машинного обучения. Признаки, которые слабо коррелируют с целевым признаком, можно попробовать исключить из построения модели, иногда это повышает качество модели. Нужно отметить, что некоторые алгоритмы машинного обучения автоматически определяют ценность того или иного признака для построения модели.
2. Понять какие нецелевые признаки линейно зависимы между собой. Линейно зависимые признаки, как правило, очень плохо влияют на качество моделей. Поэтому если несколько признаков линейно зависимы, то для построения модели из них выбирают какой-то один признак.

```
numeric_df = df.select_dtypes(include=['number'])  # Выбираем только числовые столбцы
numeric_df.corr()
```

| | Year | Duration | Time_Signature | Danceability | Energy | Key | Loudness | Speechiness | Acousticness | Instrumentalness | Liveness | Valence |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Year** | 1.000000 | -0.189393 | 0.015597 | 0.076979 | 0.049895 | 0.033728 | 0.202742 | -0.031360 | -0.017693 | -0.065800 | -0.048412 | -0.079828 |
| **Duration** | -0.189393 | 1.000000 | 0.002189 | -0.038513 | 0.137261 | -0.009721 | 0.033010 | -0.014658 | -0.120008 | 0.138951 | 0.192034 | -0.003672 |
| **Time_Signature** | 0.015597 | 0.002189 | 1.000000 | 0.198352 | 0.107461 | -0.033549 | 0.028226 | 0.067406 | -0.140952 | -0.065136 | -0.004951 | 0.095594 |
| **Danceability** | 0.076979 | -0.038513 | 0.198352 | 1.000000 | 0.306758 | -0.107458 | 0.239523 | 0.109498 | -0.228475 | -0.205886 | -0.161245 | 0.584194 |
| **Energy** | 0.049895 | 0.137261 | 0.107461 | 0.306758 | 1.000000 | 0.008558 | 0.753119 | 0.198756 | -0.670676 | -0.193241 | 0.231230 | 0.473642 |
| **Key** | 0.033728 | -0.009721 | -0.033549 | -0.107458 | 0.008558 | 1.000000 | 0.028508 | 0.047261 | 0.091315 | 0.086917 | 0.019283 | 0.022256 |
| **Loudness** | 0.202742 | 0.033010 | 0.028226 | 0.239523 | 0.753119 | 0.028508 | 1.000000 | 0.027191 | -0.460407 | -0.338593 | 0.213055 | 0.343379 |
| **Speechiness** | -0.031360 | -0.014658 | 0.067406 | 0.109498 | 0.198756 | 0.047261 | 0.027191 | 1.000000 | -0.077038 | -0.091874 | 0.175258 | 0.126809 |
| **Acousticness** | -0.017693 | -0.120008 | -0.140952 | -0.228475 | -0.670676 | 0.091315 | -0.460407 | -0.077038 | 1.000000 | 0.128933 | -0.051832 | -0.199018 |
| **Instrumentalness** | -0.065800 | 0.138951 | -0.065136 | -0.205886 | -0.193241 | 0.086917 | -0.338593 | -0.091874 | 0.128933 | 1.000000 | -0.060330 | -0.230340 |
| **Liveness** | -0.048412 | 0.192034 | -0.004951 | -0.161245 | 0.231230 | 0.019283 | 0.213055 | 0.175258 | -0.051832 | -0.060330 | 1.000000 | 0.038815 |
| **Valence** | -0.079828 | -0.003672 | 0.095594 | 0.584194 | 0.473642 | 0.022256 | 0.343379 | 0.126809 | -0.199018 | -0.230340 | 0.038815 | 1.000000 |
| **Tempo** | -0.016918 | -0.002062 | -0.147418 | 0.014639 | 0.224133 | -0.082481 | 0.171401 | 0.067646 | -0.149082 | -0.097257 | 0.111879 | 0.123279 |
| **Popularity** | 0.082522 | -0.132669 | 0.015081 | 0.158759 | -0.010802 | 0.058229 | 0.074378 | -0.128745 | 0.032508 | -0.085406 | -0.136762 | 0.134713 |

# Построим тепловую карту корреляций между числовыми признаками.

```
# Построение тепловой карты корреляций между числовыми признаками
%matplotlib inline
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Корреляционная матрица")
plt.show()
```




# Построим теперь корреляционную матрицу Пирсона.

Корреляционная матрица содержит коэффициенты корреляции между всеми парами признаков.

Корреляционная матрица симметрична относительно главной диагонали. На главной диагонали расположены единицы (корреляция признака самого с собой).

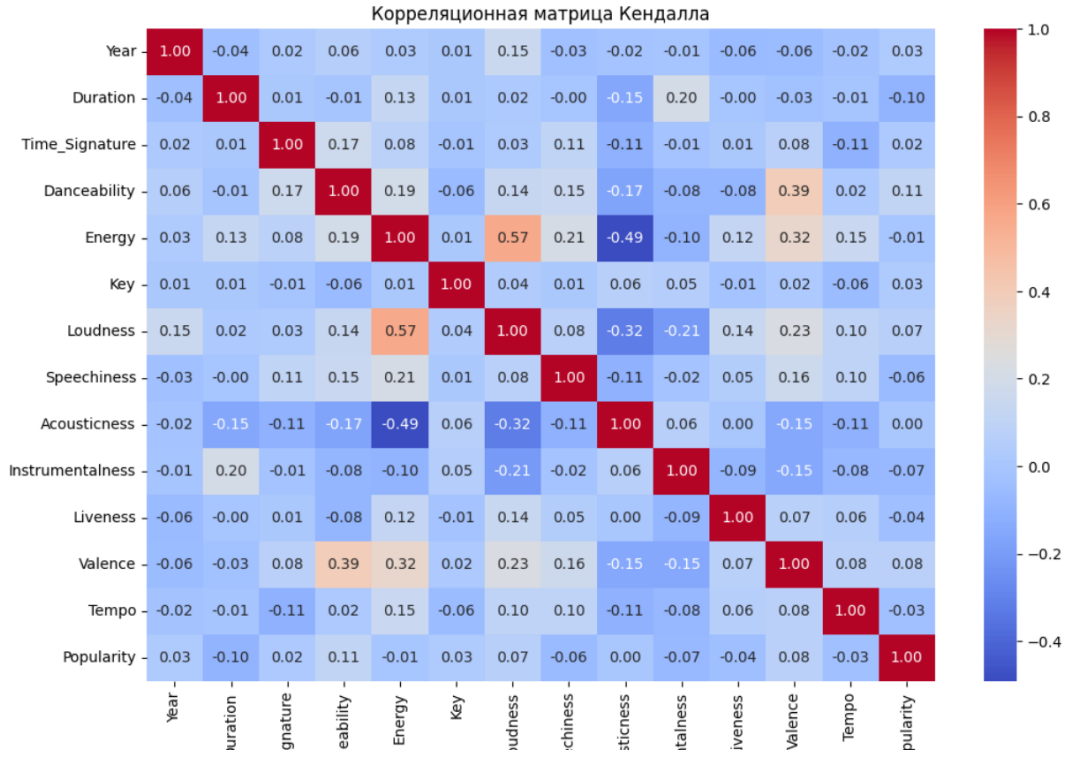
На основе корреляционной матрицы можно сделать следующие выводы: Заметим, что есть высокая положительная связь между энергичностью и громкостью трека, высокая отрицательная связь между энергичностью и акустичностью, средняя положительная связь между танцевальностью и позитивности.

По умолчанию при построении матрицы используется коэффициент корреляции Пирсона. Возможно также построить корреляционную матрицу на основе коэффициентов корреляции Кендалла и Спирмена. На практике три метода редко дают значимые различия.

```
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True, method='pearson'), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Корреляционная матрица Пирсона")
plt.show()
```

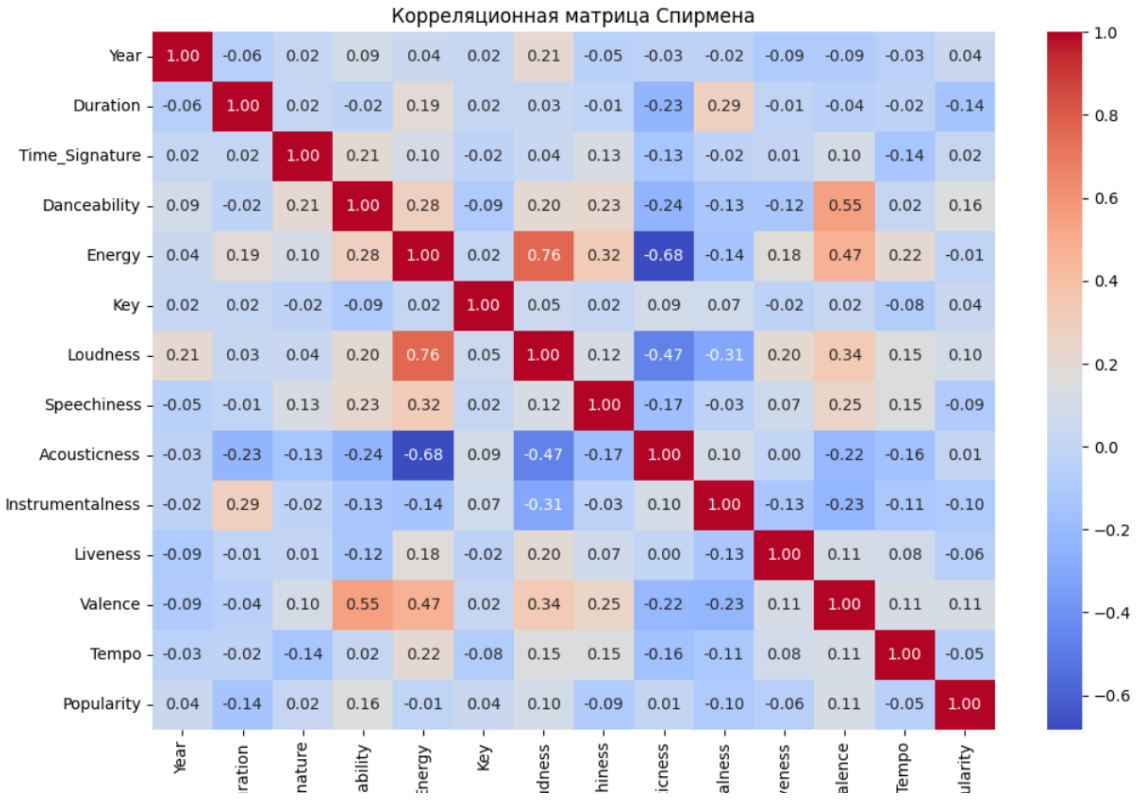
Построим корреляционную матрицу Кендалла.

```
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True, method='kendall'), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Корреляционная матрица Кендалла")
plt.show()
```



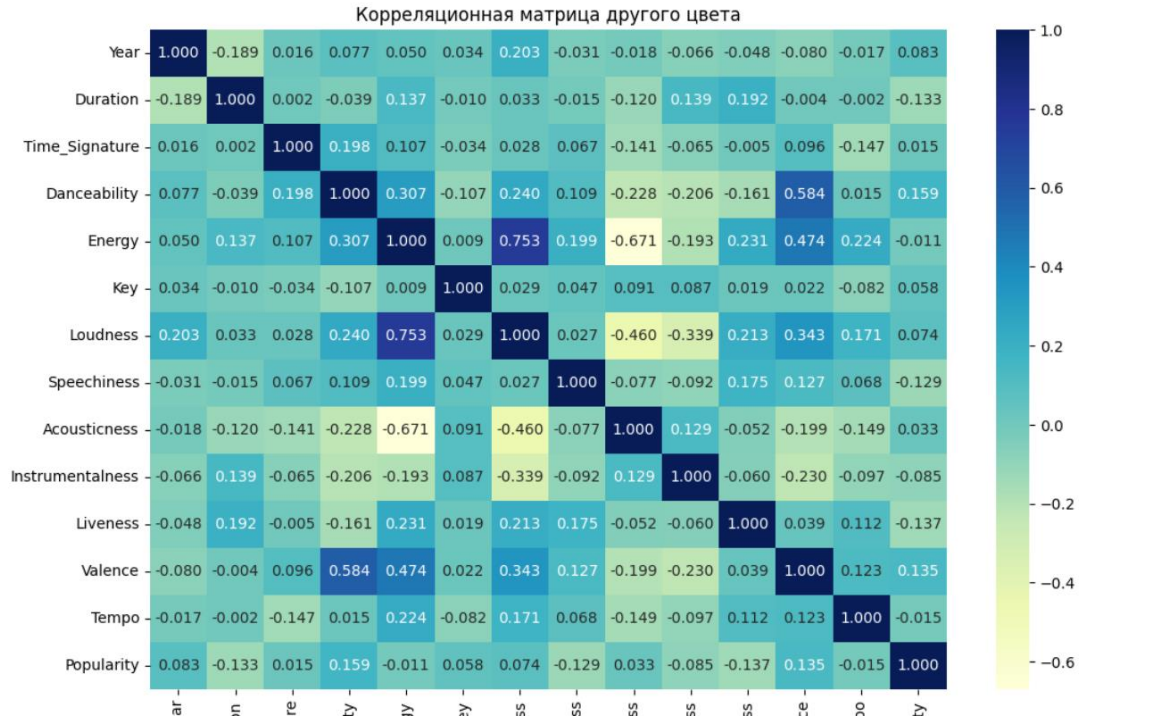

Построим корреляционную матрицу Спирмена.

```
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True, method='spearman'), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Корреляционная матрица Спирмена")
plt.show()
```



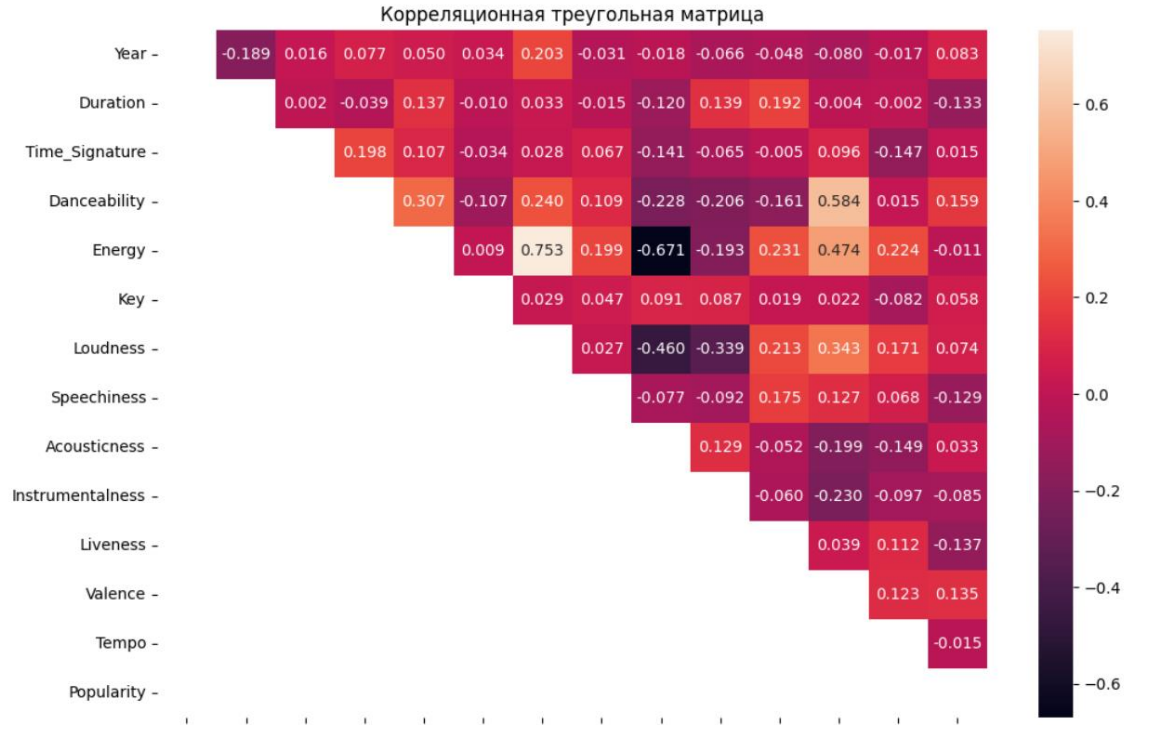
Построим корреляционную матрицу другой цветовой гаммы.

```
# Изменение цветовой гаммы
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True), cmap='YlGnBu', annot=True, fmt='.3f')
plt.title("Корреляционная матрица другого цвета")
plt.show()
```




Построим треугольную корреляционную матрицу.

```
# Треугольный вариант матрицы
mask = np.zeros_like(df.corr(numeric_only=True), dtype=np.bool)
# чтобы оставить нижнюю часть матрицы
# mask[np.triu_indices_from(mask)] = True
# чтобы оставить верхнюю часть матрицы
plt.figure(figsize=(12, 8))
mask[np.tril_indices_from(mask)] = True
sns.heatmap(df.corr(numeric_only=True), mask=mask, annot=True, fmt='.3f')
plt.title("Корреляционная треугольная матрица")
plt.show()
```



Построим корреляционные матрицы различными методами.

```
fig, ax = plt.subplots(1, 2, sharex='col', sharey='row', figsize=(22,7))
sns.heatmap(df.corr(numeric_only=True, method='pearson'), ax=ax[0], annot=True, fmt='.2f')
sns.heatmap(df.corr(numeric_only=True, method='kendall'), ax=ax[1], annot=True, fmt='.2f')
fig.suptitle('Корреляционные матрицы, построенные различными методами')
ax[0].title.set_text('Pearson')
ax[1].title.set_text('Kendall')
```