

Рекомендательные системы по подбору музыкальных треков с использованием методов ML

Райзанат Шурпаева
Студентка Нетологии



План работы

1

Вступление. Знакомство с данными, первичная обработка, визуализация.

2

Разработка функций. Нормализация данных, векторизация TD-IDF, расчет метрики RMSE.

3

Построение рекомендательной системы методом вычисления манхэттенских расстояний. Вариант с добавлением пользователей и рейтинга.

4

Заключение.



В настоящее время существует несколько крупных музыкальных стриминговых сервисов - Apple Music, SoundCloud, Spotify и др. Из российских – VK музыка, Яндекс.Музыка.

Вне зависимости от того, какую музыку предпочитают люди, они сталкиваются с одной и той же проблемой, когда личная фонотека заслушана и хочется чего-нибудь новенького. С одной стороны, выбор огромен — мир музыки очень богат, он насчитывает сотни миллионов треков и пополняется каждый день. С другой стороны, сориентироваться в этом разнообразии бывает непросто: музыки много, а человек один. Собственно, **целью** моей работы является решение этой проблемы.

Для данного проекта были использованы данные мирового стримингового сервиса Spotify. Если верить слушателям, его алгоритмы рекомендаций считаются самыми удачными.



Ссылка на датасет: <https://www.kaggle.com/datasets/zaheenhamidani/ultimate-spotify-tracks-db>



Признаки

1. Genre – жанр;

2. Artist_name – имя артиста;

3. Track_name – название трека;

4. Track_id – id трека;

5. Popularity – популярность;

6. Acousticness – акустичность;

7. Danceability – танцевальность;

8. Duration_ms – продолжительность;

9. Energy – энергия;
10. Instrumentalness – инструментальность;

11. Key – ключ;

12. Liveness – живучесть;

13. Loudness – громкость;

14. Mode – модальность;

15. Speechiness – безсловесный;

16. Tempo – темп;

17. Time_signature – такт;

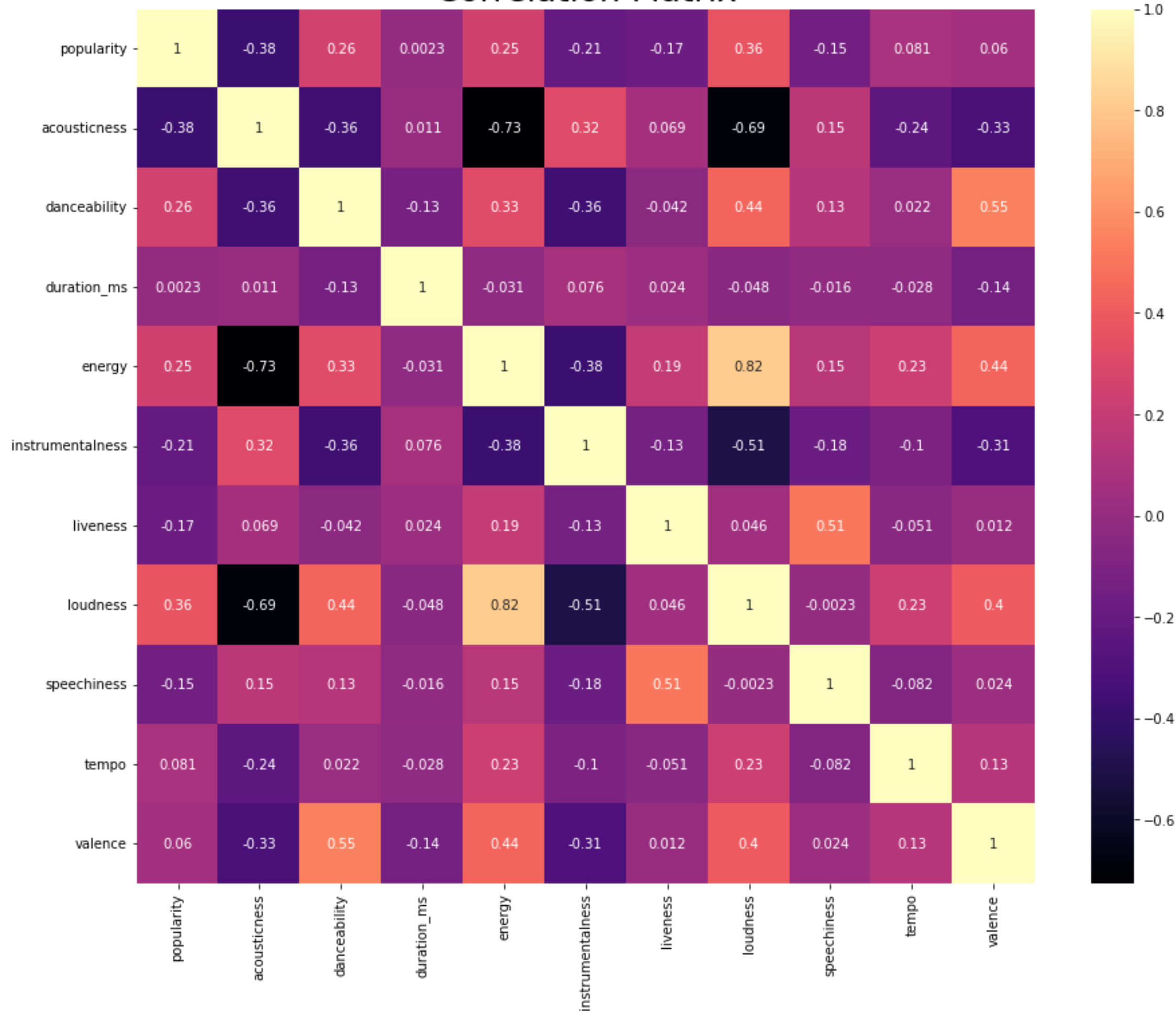
18. Valence – валентность.

data = pd.read_csv("SpotifyFeatures.csv")
data

genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	tim
Movie	Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.61100	0.389	99373	0.910	0.000000	C#	0.3460	-1.828	Major	0.0525	166.969	
Movie	Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.24600	0.590	137373	0.737	0.000000	F#	0.1510	-5.559	Minor	0.0868	174.003	
Movie	Joseph Williams	Don't Let Me Be Lonely Tonight	0CoSDzoNIKCRs124s9uTVy	3	0.95200	0.663	170267	0.131	0.000000	C	0.1030	-13.879	Minor	0.0362	99.488	



Correlation Matrix

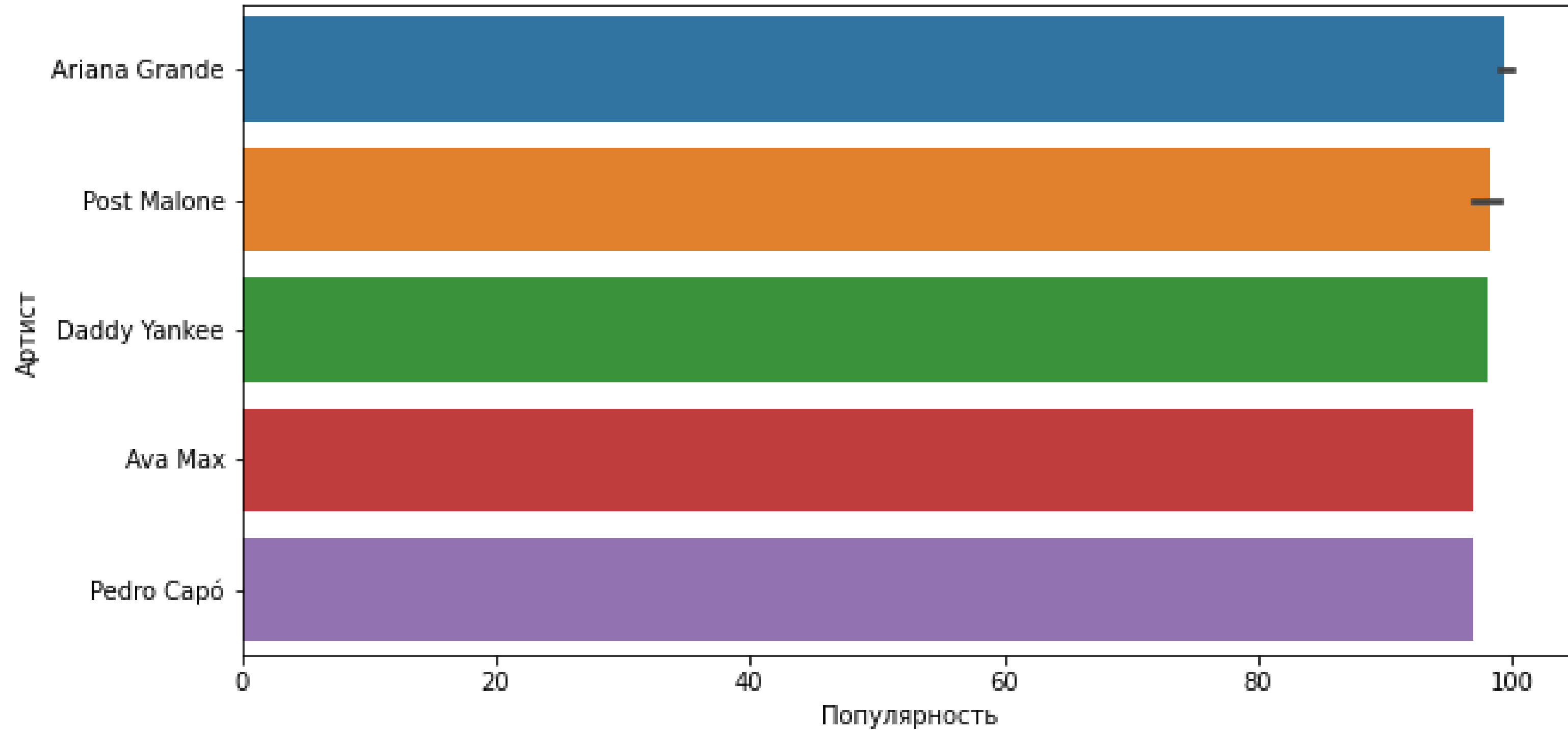


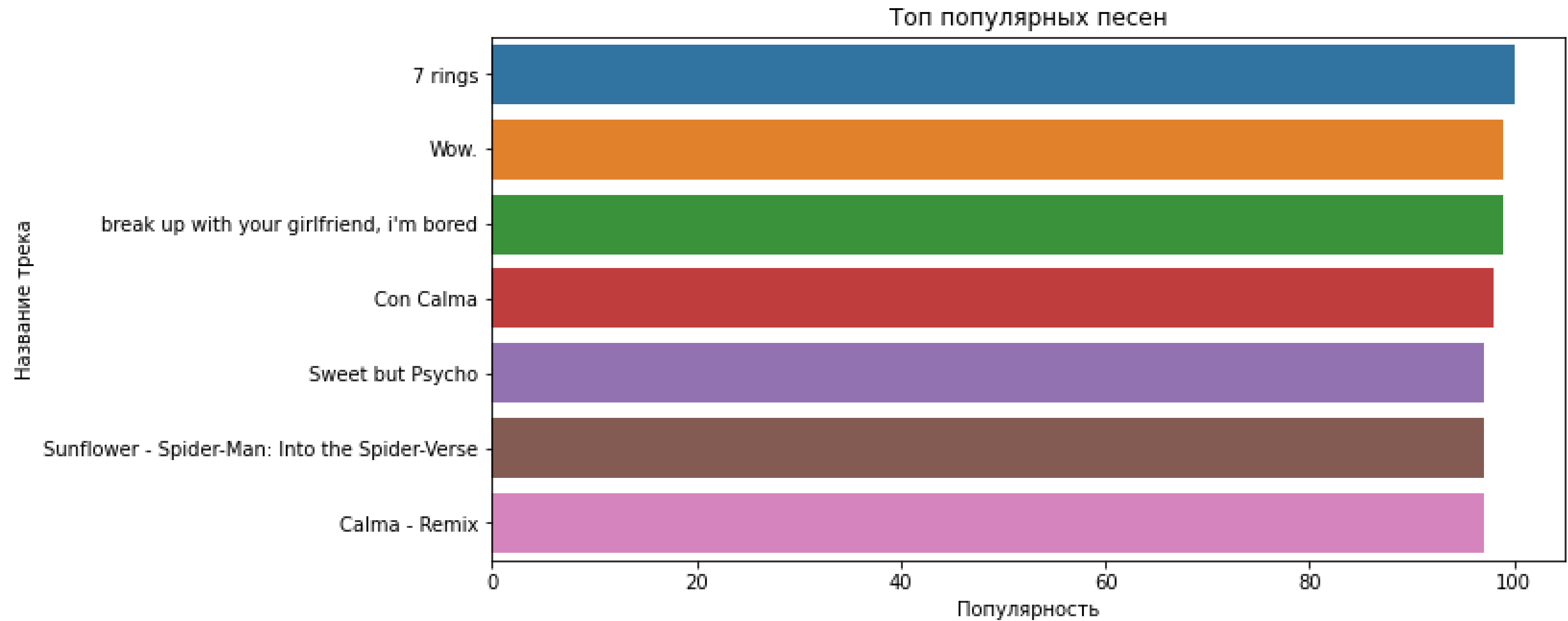
В данных
отсутствуют
пустые значения.

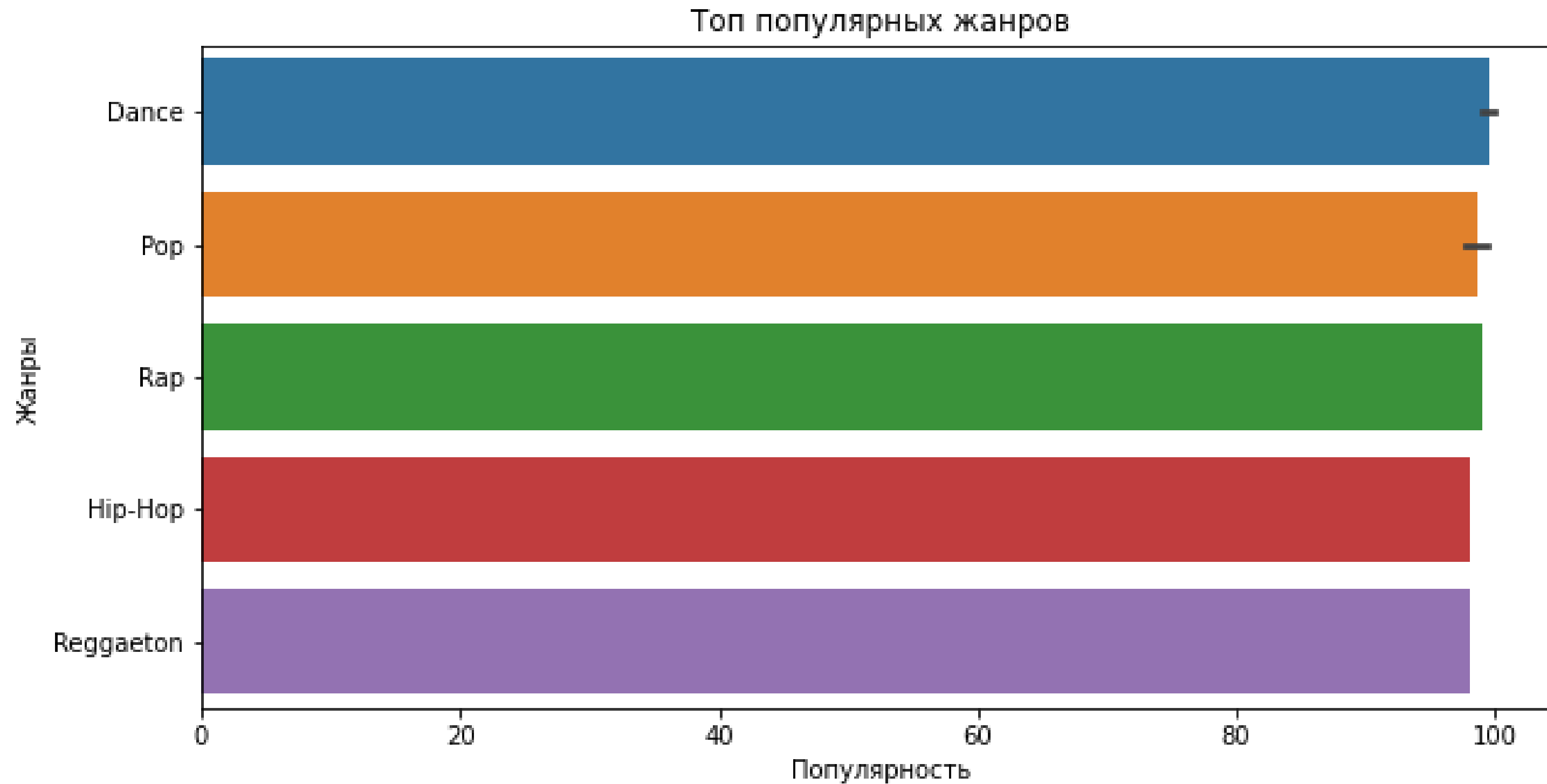
По матрице мы
видим, что больше
всего между собой
коррелируют
энергия –
акустичность,
громкость –
акустичность,
громкость - энергия

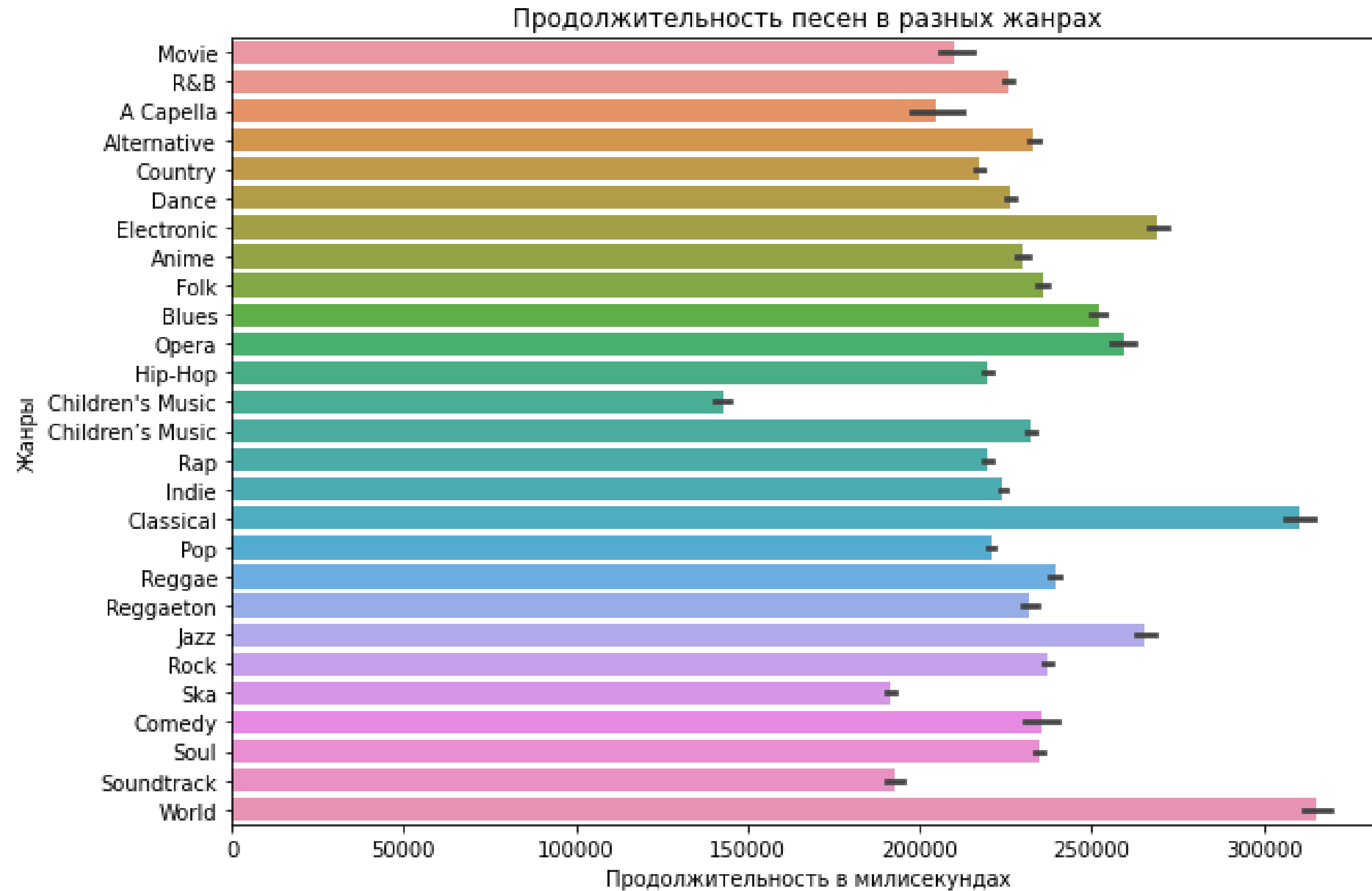


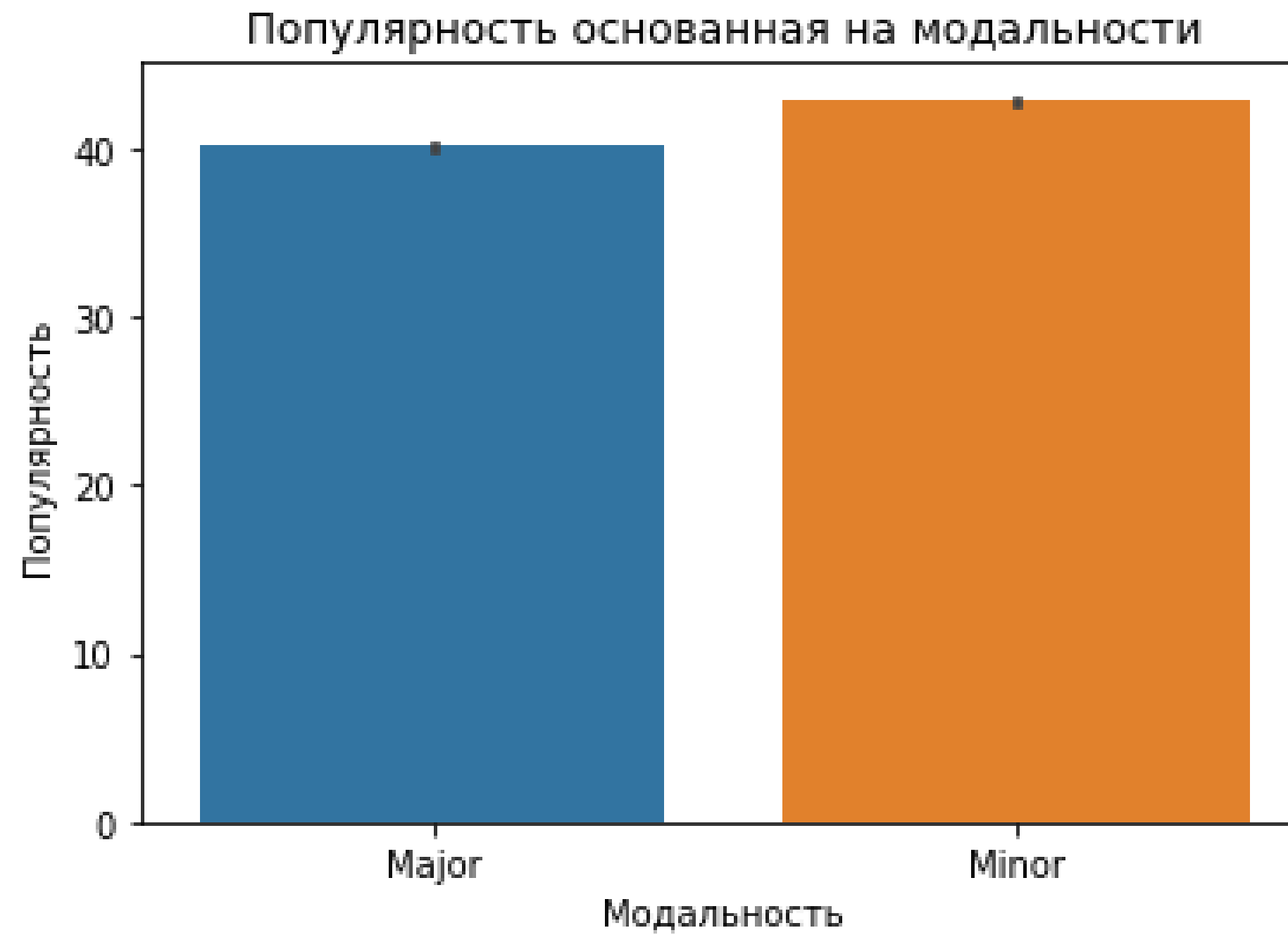
Топ популярных артистов











Обрабатываем данные, применяем **StandardScaler** (метод, который стандартизирует диапазон функциональных возможностей входного набора данных).

Преобразовываем жанры, векторизуем с помощью **TD-IDF** (от англ. TF — term frequency, IDF — inverse document frequency) — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса.

```
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(df.genre)
genres = tfidf.get_feature_names()
tfidf_matrix = pd.DataFrame(tfidf_matrix.toarray(), columns=genres)
tfidf_matrix.head()
```

`/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 a warnings.warn(msg, category=FutureWarning)`

	alternative	anime	blues	capella	children	classical	comedy	country	dance	electronic	...	opera	pop	rap	reggae	reggaeton	rock	ska	soul	soundtrack	world
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 27 columns



Строим линейную регрессию и оцениваем метрику **RMSE**
(Среднеквадратическая ошибка) на тестовой выборке

```
[ ] model = LinearRegression()  
    model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
[ ] y_pred = model.predict(X_test)
```

```
▶ MSE = mean_squared_error(y_test, y_pred)  
MSE
```

```
↳ 0.3955523234620444
```

```
[ ] RMSE = math.sqrt(MSE)  
    print(RMSE)
```

```
0.628929505955989
```

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} = \sqrt{\text{MSE}}$$



Поскольку мы будем использовать наши данные для вычисления манхэттенских расстояний* между песнями, создадим функцию для их нормализации.

Отдельно нормализуем все числовые столбцы.

```
[85] def normalize_column(col):  
      max_d = data[col].max()  
      min_d = data[col].min()  
      data[col] = (data[col] - min_d)/(max_d - min_d)
```


```
[86] num_types = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']  
      num = data.select_dtypes(include=num_types)  
  
      for col in num.columns:  
          normalize_column(col)
```

*Расстояние городских кварталов (манхэттенское расстояние)
Это расстояние является средним разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида. Однако для этой меры влияние отдельных больших разностей (выбросов) уменьшается (т.к. они не возводятся в квадрат). Формула для расчета манхэттенского расстояния:

$$\rho(x, x') = \sum_i^N |x_i - x'_i|$$



Существует вероятность, что песни из разных жанров могут иметь довольно схожие характеристики, и это нехорошо. Поэтому мы создадим новую функцию, которая отличала бы песни разных групп. Для этой цели мы будем использовать кластеризацию К-средних* с 10 кластерами.



```
from sklearn.cluster import KMeans
km = KMeans(n_clusters=10)
predicted = km.fit_predict(num)
data['predicted'] = predicted
normalize_column('predicted')
```

*Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров.



Рекомендательная система методом вычисления манхэттенских расстояний между треками

```
class SongRecommender():
    def __init__(self, rec_data):
        self.rec_data_ = rec_data

    #если нам нужно изменить данные
    def change_data(self, rec_data):
        self.rec_data_ = rec_data

    #функция, которая возвращает рекомендации, мы также можем выбрать количество песен, которые будут рекомендованы
    def get_recommendations(self, song_name, amount=1):
        distances = []
        #выбираем данные для нашей песни
        song = self.rec_data_[self.rec_data_.track_name.str.lower() == song_name.lower()].head(1).values[0]
        #сброс данных с помощью нашей песни
        res_data = self.rec_data_[self.rec_data_.track_name.str.lower() != song_name.lower()]
        for r_song in tqdm(res_data.values):
            dist = 0
            for col in np.arange(len(res_data.columns)):
                #индексы нечисловых столбцов
                if not col in [0,1,2]:
                    #вычисление манхэттенских расстояний для каждого числового признака
                    dist = dist + np.absolute(float(song[col]) - float(r_song[col]))
            distances.append(dist)
        res_data['distance'] = distances
        #сортировка наших данных по возрастанию по функции "расстояние"
        res_data = res_data.sort_values('distance')
        columns = ['genre', 'artist_name', 'track_name', 'distance']
        return res_data[columns][:amount]
```

```
[135] recommender = SongRecommender(data)
```



Выводим список рекомендованных треков, с указанием дистанции от меньшего к большему

```
recommender.get_recommendations('Pyramids', 10)
```

```
100%|██████████| 191051/191051 [00:06<00:00, 28618.73it/s]  
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:24: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

	artist_name	track_name	distance
207710	Jesus Culture	Revelation Song - Live	0.289372
121658	Christine D'Clario	Que Se Abra el Cielo	0.408868
207449	Elevation Worship	Yours (Glory and Praise)	0.410361
207752	Israel Houghton	Jesus At the Center - Live	0.434873
207636	En Espíritu Y En Verdad	Cuan Grande Es Dios	0.457691
79007	Michael W. Smith	Waymaker	0.467847
217165	Hillsong UNITED	Take Heart - Live	0.467990
208841	Bethel Music	Cornerstone (Spontaneous)	0.491786
208176	Jesus Culture	Everything And Nothing Less - Live	0.500828
208063	Israel & New Breed	Tu presencia es el Cielo	0.517223



Вариант рекомендательной системы с добавлением пользователей и рейтинга

```
[149] spotify['user_id'] = np.random.randint(1000,2000,len(spotify))
spotify['rating'] = np.random.randint(1,6,len(spotify))
spotify.head()
```

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	t:
0	Movie	Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.611	0.389	99373	0.910	0.000	C#	0.3460	-1.828	Major	0.0525	166.969	
1	Movie	Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.246	0.590	137373	0.737	0.000	F#	0.1510	-5.559	Minor	0.0868	174.003	
2	Movie	Joseph Williams	Don't Let Me Be Lonely Tonight	0CoSDzoNIKCRs124s9uTVy	3	0.952	0.663	170267	0.131	0.000	C	0.1030	-13.879	Minor	0.0362	99.488	
3	Movie	Henri Salvador	Dis-moi Monsieur Gordon Cooper	0Gc6TVm52BwZD07Ki6tlvf	0	0.703	0.240	152427	0.326	0.000	C#	0.0985	-12.178	Major	0.0395	171.758	
4	Movie	Fabien Nataf	Ouverture	0lusIXpMROHdEPvSI1fTQK	4	0.950	0.331	82625	0.225	0.123	F	0.2020	-21.150	Major	0.0456	140.576	



Преобразуем некоторые категориальные данные в числовые и построим рекомендацию:

```
[196] from sklearn.preprocessing import LabelEncoder
      le = LabelEncoder()
      le.fit(spotify['mode'])
      spotify['mode_le']=le.transform(spotify['mode'])
```

```
[197] le = LabelEncoder()
      le.fit(spotify['genre'])
      spotify['genre_le']=le.transform(spotify['genre'])
```

```
▶ def recommender(user):

    fav_genre = spotify[spotify['user_id']==user].sort_values(by=['rating', 'mode_le', 'popularity', 'genre_le'], ascending=False)['genre'][:1]
    fav_genre = list(dict.fromkeys(fav_genre))

    # удаляем треки из списков, которые были прослушаны ранее
    listened_track = spotify.index[spotify['track_name'].isin(['2009', 'Pyramids', 'Coldest Winter'])].tolist()

    # треки, еще не обнаруженные пользователем
    remaining_track = spotify.drop(listened_track, axis=0)
    CanBeRecommended = remaining_track[remaining_track['genre'].isin(fav_genre)]

    # сортировка треков в любимом жанре пользователя
    CanBeRecommended = CanBeRecommended.sort_values(by=['rating', ], ascending=False)[['track_name', 'artist_name', 'genre', 'rating', ]][:10]

    return CanBeRecommended
```



Рекомендации для отдельного пользователя

Рекомендация в одном прослушивающем жанре

recommender(1251)

	track_name	artist_name	genre	rating
150749	What a Girl Likes	Cardi B	Pop	5
109561	Really Really	Kevin Gates	Pop	5
112822	Chateau	Angus & Julia Stone	Pop	5
112818	Nervous - The Ooh Song/Mark McCabe Remix	Gavin James	Pop	5
109554	A Little Too Much	Shawn Mendes	Pop	5
112815	Fuck Nigga	T.I.	Pop	5
112809	Don't You Want Me	The Human League	Pop	5
112807	If I Lose Myself - Alesso vs OneRepublic	OneRepublic	Pop	5
109563	So It Goes	Mac Miller	Pop	5
109548	Shake It Out	Florence + The Machine	Pop	5

Рекомендация в пяти близких жанрах

recommender(1251)

	track_name	artist_name	genre	rating
141567	Glue	Fickle Friends	Indie	5
140849	Paranoia in B Major	The Avett Brothers	Indie	5
111455	Obsessed	Mariah Carey	Pop	5
87109	I Don't Wanna Do This Anymore	XXXTENTACION	Rap	5
140853	Honey Slider	Houndmouth	Indie	5
87112	Pretty Little Fears (feat. J. Cole)	6LACK	Rap	5
87113	Nights Like This (feat. Ty Dolla \$ign)	Kehlani	Rap	5
140852	Wash It All Away	San Cisco	Indie	5
114927	Buttons	Mac Miller	Rap	5
87118	Foot Fungus	Ski Mask The Slump God	Rap	5



Заключение

В работе была разработана модель рекомендательной системы для музыки на стриминговом сервисе. В ходе работы проанализирован исходный датасет, данные проверены на наличие пустых значений, выявлены дубликаты.

Для построения рекомендательной системы выбран Item-based подход, так как датасет имеет числовые характеристики для каждого трека (акустичность, темп, модальность и т.д.). Эти данные использованы, чтобы найти наиболее похожие песни с расчетом дистанции между ними.

Чтобы найти разницу между песнями, рассчитано расстояние по Манхэттену между всеми из них. И, как результат, в рекомендации попадают песни с наименьшей дистанцией.

Также в работе применена векторизация жанров TD-IDF, рассчитан RMSE $\sim 0,62$

Дополнительно добавлены пользователи и рейтинг песен, построена рекомендация для отдельного пользователя в рамках одного жанра и в пяти ближайших.

Код проекта доступен по ссылке

https://github.com/shurpaeva/Recsis_Diplom_DS/blob/main/Рекомендательные_системы_по_подбору_музыкальных_треков_с_использованием_методов_ML.ipynb

