A Data Science Project : Music Recommendation System

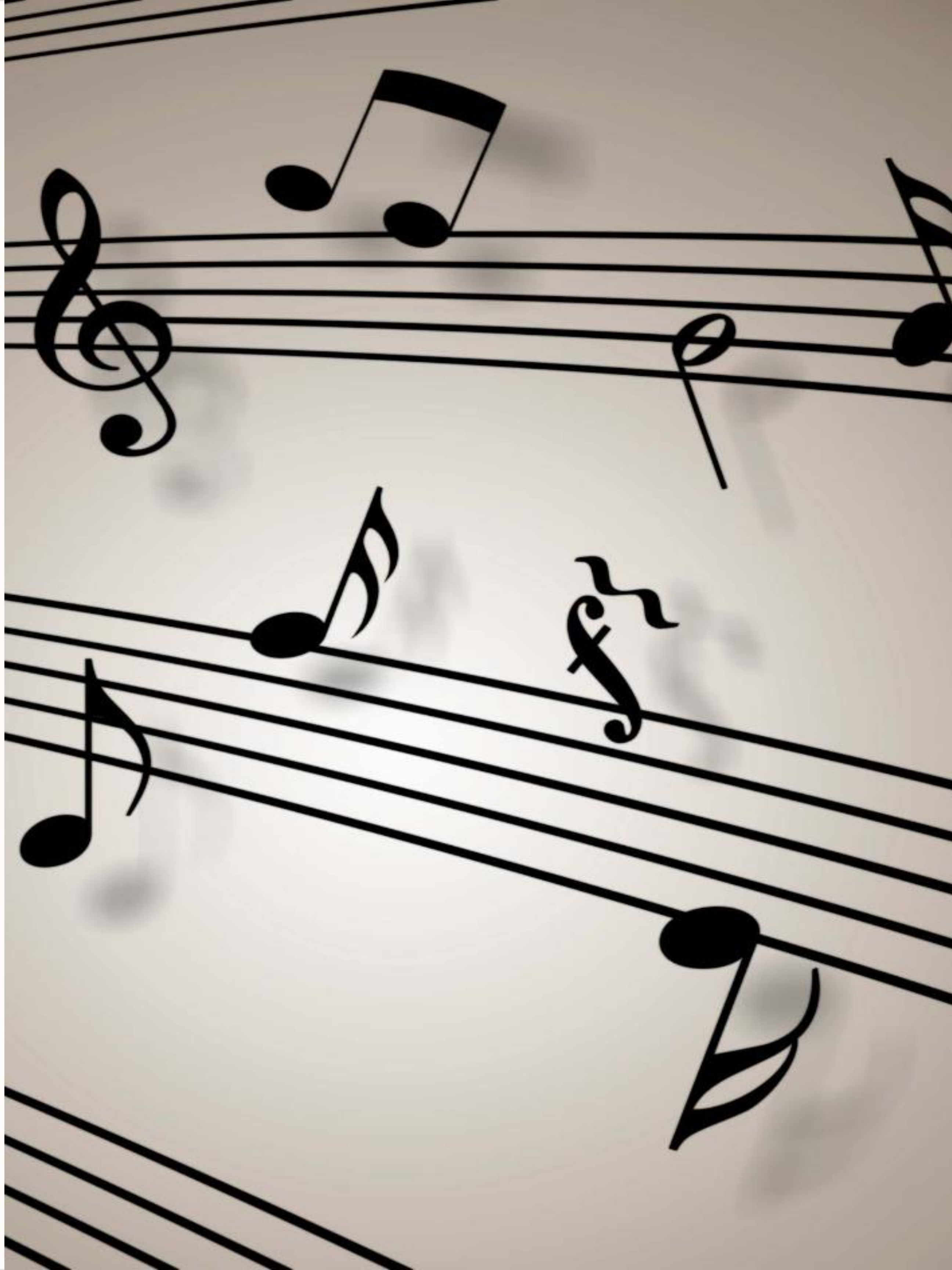
Discover the power of music recommendation systems and learn how we developed a **cutting-edge system** to enhance **user experience** and **satisfaction**. Discover new tunes, **create your perfect playlists**, and indulge in musical bliss.

Team Members:

- Shivangi Thakur
- Pavan Raju
- Sanjeev Raj
- Anand Venkata
- Abhishek Warfade
- Prajakta Kutal

Mentor:

- Neha Ramchandani



Why Music Recommendation? Recommendation?

1 Information overload

The amount of music available is immense and can be overwhelming.

2 Personalization 🎧

Everyone has a unique taste in music, and recommendation engines can cater to them.

3 Discovery 🔍

Recommendations can lead music lovers to discover new genres, artists, genres, artists, and tracks they might have missed otherwise.

Introduction

What is a music recommendation system?

Explore the concept and functionalities of music recommendation systems.

Importance of music recommendation systems

Understand the impact of personalized music recommendations on user engagement and retention.

Overview of the project

Get an overview of our data data science project on building a music recommendation system.

Know more about Music Recommender System:

- The Recommender System is a software application and algorithm that provides suggestions for items that a user is most interested in.
- Recommendations are used in a variety of real-world situations, such as deciding what products to buy, listening to music, or reading the latest news.
- Now talking about the Music Recommendation System, the availability of digital music is now abundant due to the new business model in the music industry.
- As a result, the importance of a music recommender system for music suppliers cannot be overstated. It is foreseeing.
- Collaborative filtering makes suggestions based on the collaborative power of the available evaluation by users.
- It is assumed that if people rate music things similarly or behave similarly, they would rate other music items similarly as well.

Importance of Music Recommendation System:

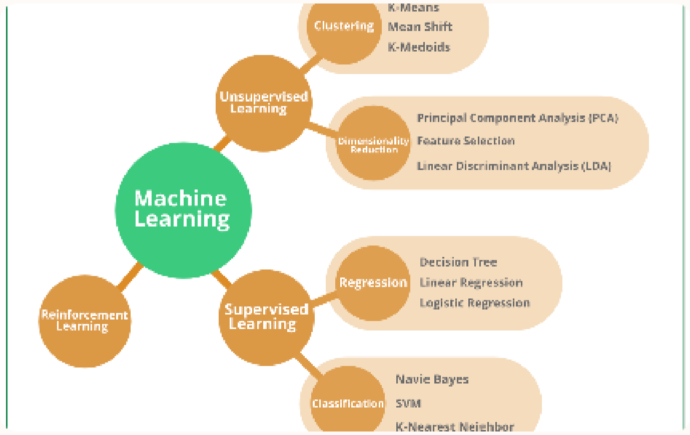
- Provide quality and immersive customer streaming experience
- Make your platform maximum personalized
- Increase satisfaction and engagement of your customers
- Make it convenient to use your service, with no need to waste time finding new songs
- Automate curating and playlisting audio
- Get insights about users' behavior and make data-based marketing decisions.

Overview:

- With the rise of digital content distribution, we have access to a huge music collection. With millions of songs to choose from, we sometimes feel overwhelmed. Thus, an efficient music recommender system is necessary in the interest of both music service providers and customers.

- Our music recommender system is large-scale and personalized. We learn from users' listening history and features of songs and predict songs that a user would like to listen to.

How It Works



Data Analysis

The system collects data from various sources such as music streaming platforms, social media, media, and user preferences.

Algorithm

Using sophisticated machine learning techniques, the algorithms learn about the user's preferences and suggest similar music.

Application

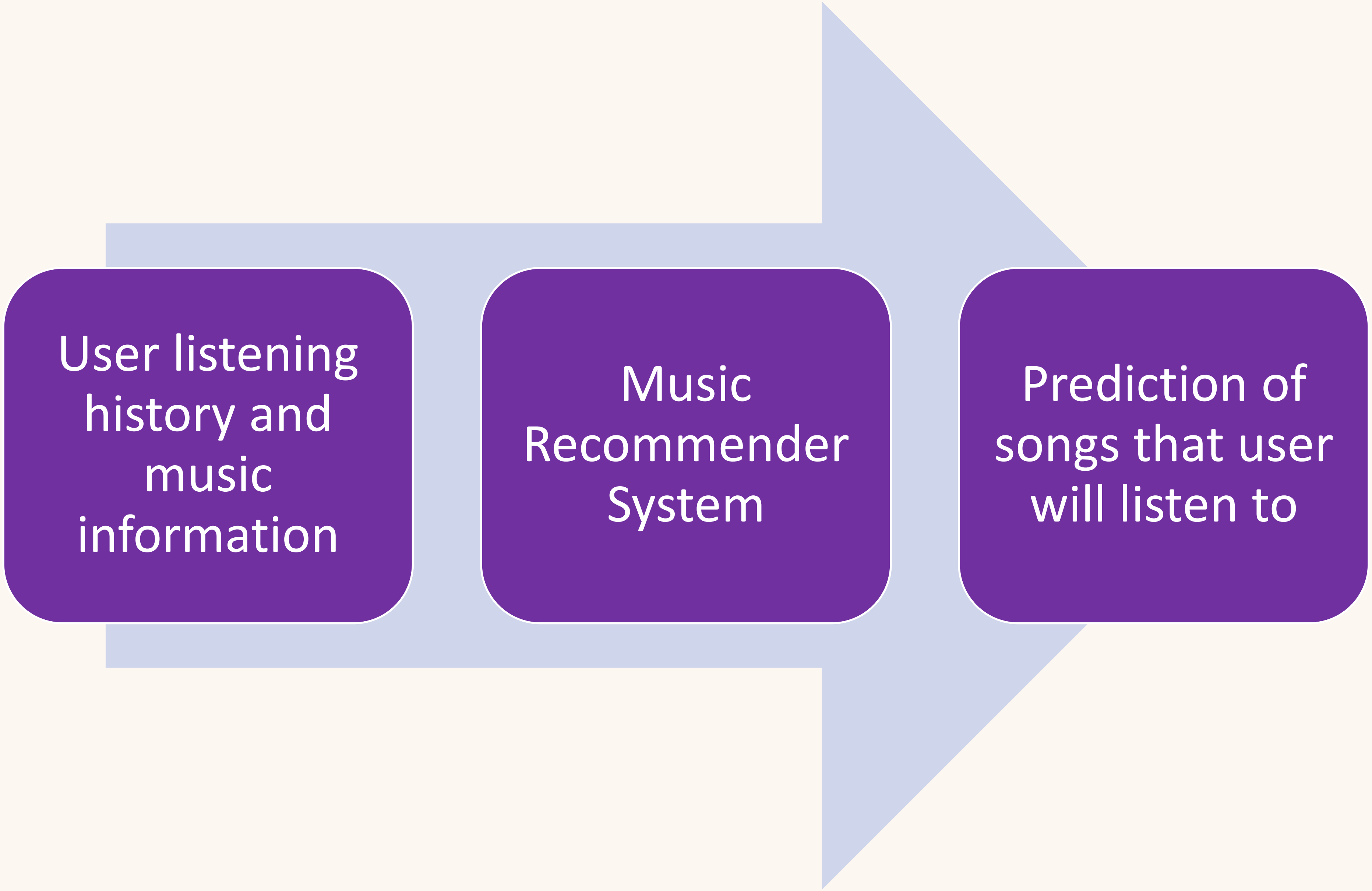
The system integrates with music streaming applications like Spotify or Apple Music and recommends personalized playlists to the user.

Project Workflow:

- Understanding the objectives
- Importing the necessary libraries
- Loading the Dataset
- Data Understanding
- Data preprocessing
- Exploratory Data Analysis (EDA)
- Feature Scaling
- Data Visualization
- Model Building using User Based Collaborative Filtering(UBCF)
- Model Training and Testing
- Model Deployment

Project Objective:

To build a feature of recommendation system to support a music application



How does the system work?

1

Data Collection & Analysis

Massive amounts of data are collected from various sources, including user listening habits, artist information, and social media trends.

2

Recommendation Algorithms

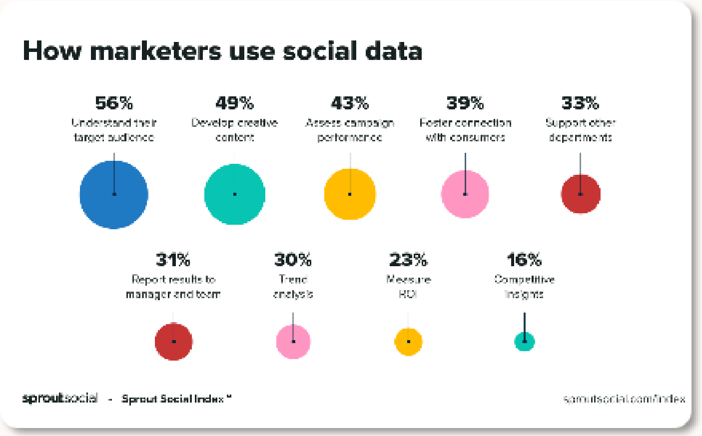
Sophisticated algorithms analyze the collected data to generate personalized personalized recommendations based on based on user preferences, genre preferences, and more.

3

User Interface

The user interface of the music recommendation system presents the recommendations in an intuitive and user-user-friendly way for easy exploration and and discovery.

Data collection and analysis



User Listening Habits

Analyzing user listening habits habits helps determine musical musical preferences, favorite genres, and frequently played artists.

Social Media Trends

By monitoring social media platforms, the system captures emerging trends and identifies popular songs, artists, and genres.

Artist Information

Collecting data about individual individual artists, their style, collaborations, and attributes helps in creating meaningful recommendations.

Data Collection and Analysis

1

Data Collection

The music streaming platforms are the primary data source for the recommendation engine. The system also collects data from social media, user searches, and purchase history.

2

Pre-processing

The data might contain noise and irrelevant metadata. Pre-processing the processing the data using techniques techniques such as feature scaling, normalization, and outlier removal helps clean the data.

3

Analysis

Statistical analysis and data visualization visualization techniques such as PCA are PCA are used to understand the underlying patterns and correlations in correlations in the data. This helps in in getting better recommendations. recommendations.

4

Feature extraction

Explore the process of extracting key features from features from the music data for effective recommendations.

After understanding the objective of the project and then importing all the necessary libraries, we will import the required dataset in csv format. Our dataset consist of 7001 rows and 21 columns in total.

	user_id	song_id	spotify_popularity	track_name	duration_ms	explicit	danceability	energy	key	loudness	...	speechiness	acousticness	instrum
0	3720277.0	32192.0	87.0	Comedy	230666.0	False	0.676	0.4610	1.0	-6.746	...	0.1430	0.032200	
1	3720277.0	6801.0	45.0	Ghost - Acoustic	149610.0	False	0.420	0.1660	1.0	-17.235	...	0.0763	0.924000	
2	3720277.0	31643.0	62.0	To Begin Again	210826.0	False	0.438	0.3590	0.0	-9.734	...	0.0557	0.210000	
3	3720277.0	1864239.0	72.0	Can't Help Falling In Love	201933.0	False	0.266	0.0596	0.0	-18.515	...	0.0363	0.905000	
4	3720277.0	38804.0	73.0	Hold On	198853.0	False	0.618	0.4430	2.0	-9.681	...	0.0526	0.469000	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
6996	3728657.0	1822821.0	61.0	Phantoms of Mortem Tales	335293.0	False	0.128	0.9540	5.0	-4.753	...	0.0567	0.000004	
6997	3728700.0	4222632.0	62.0	Bergagasten	310213.0	False	0.565	0.8520	0.0	-3.869	...	0.0340	0.001290	
6998	3728700.0	19250.0	53.0	The Pentagram Burns	384946.0	False	0.406	0.9770	2.0	-10.303	...	0.0809	0.007940	
6999	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
7000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

7001 rows × 21 columns

Data Understanding:

Columns/Features of our dataset

```
0 user_id
1 song_id
2 spotify_popularity
3 track_name
4 danceability
5 energy
6 key
7 mode
8 acousticness
9 instrumentalness
10 valence
11 tempo
12 time_signature
13 track_genre
14 Rating
```

Total count

```
user_id               6997
song_id               6997
spotify_popularity    6997
track_name            6997
danceability          6997
energy                6997
key                   6997
mode                  6997
acousticness          6997
instrumentalness      6997
valence               6997
tempo                 6997
time_signature        6997
track_genre           6997
Rating                6798
dtype: int64
```

Data type

```
1  Music_data.dtypes
```

```
user_id               float64
song_id               float64
spotify_popularity    float64
track_name             object
danceability          float64
energy                float64
key                   float64
mode                  float64
acousticness          float64
instrumentalness      float64
valence               float64
tempo                 float64
time_signature        float64
track_genre            object
Rating                float64
dtype: object
```

Data understanding continues...

```
#Number of unique users.

Music_data['user_id'].nunique()
```

```
665
```

```
Music_data['track_name'].value_counts()
```

```
All Star                                  18
killer                                    15
Living Dead Girl                          14
Don't Shoot Me Santa                      12
Dragula                                   12
                                          ..
Très magnifique                            1
Freguês da Meia Noite (Instrumental)       1
Follow me                                  1
African Man                                1
The Pentagram Burns                        1
Name: track_name, Length: 5656, dtype: int64
```

Shape of dataset

```
Music_data.shape
```

```
(7001, 15)
```

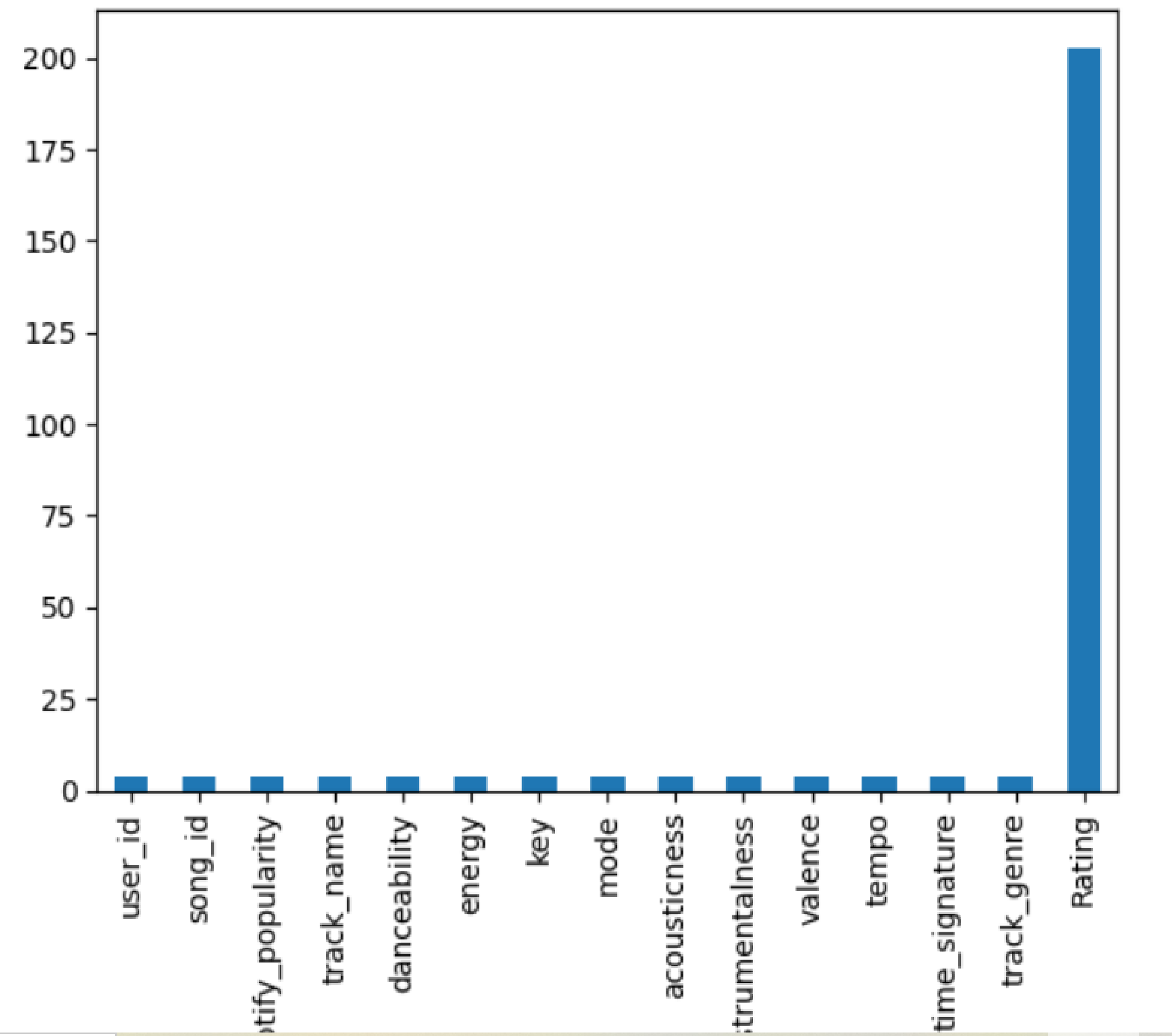
```
Music_data['track_genre'].unique()
```

```
array(['acoustic', nan, 'afrobeat', 'alt-rock', 'alternative', 'ambient',
       'anime', 'black-metal'], dtype=object)
```

Exploratory Data Analysis (EDA) || Data Preprocessing

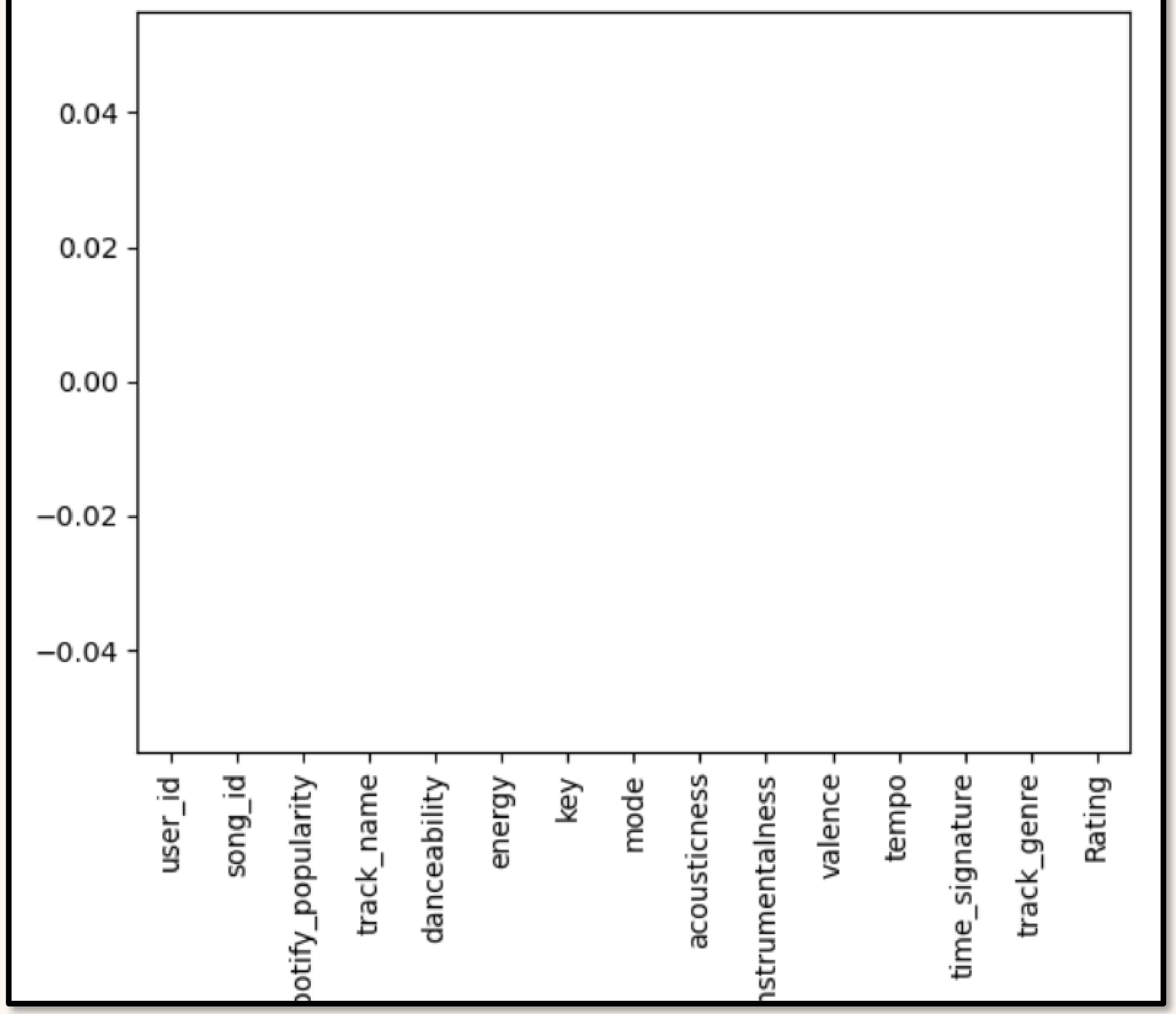
Finding out the null entries

```
Music_data.isnull().sum().plot.bar()
plt.show()
```



Removing all the null entries

```
Music_data.isnull().sum().plot.bar()
plt.show()
```



After removing all the three duplicate datapoints.

```
#CHECK THE DUPLICATES
Music_data.duplicated().sum()
```

```
3
```

```
Music_data = Music_data.drop_duplicates()
```

```
Music_data.duplicated().sum()
```

```
0
```

Describing the datasets

```
Music_data.describe()
```

	user_id	song_id	spotify_popularity	danceability	energy	key	mode	acousticness	instrumentalness	valence	
count	6.998000e+03	6.998000e+03	6998.000000	6998.000000	6998.000000	6998.000000	6998.000000	6998.000000	6998.000000	6998.000000	6
mean	3.724545e+06	2.040881e+06	50.222667	0.502244	0.628266	5.363013	0.612977	0.311448	0.251876	0.418645	
std	4.463748e+04	1.833838e+06	16.610422	0.183160	0.288064	3.551575	0.487069	0.359503	0.369823	0.268494	
min	3.324765e+02	1.200000e+02	0.000000	0.000000	0.001440	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	3.722204e+06	7.938300e+04	41.000000	0.381000	0.406000	2.000000	0.000000	0.005215	0.000000	0.185000	
50%	3.725674e+06	1.841530e+06	52.000000	0.516000	0.700000	6.000000	1.000000	0.113000	0.001140	0.382000	
75%	3.728205e+06	4.139380e+06	62.000000	0.638000	0.882000	9.000000	1.000000	0.639750	0.627000	0.637000	
max	3.734809e+06	5.336437e+06	98.000000	0.974000	1.000000	11.000000	1.000000	0.996000	0.993000	0.995000	

Dataset Information

```
#DATASET INFORMATION
Music_data.info()
```

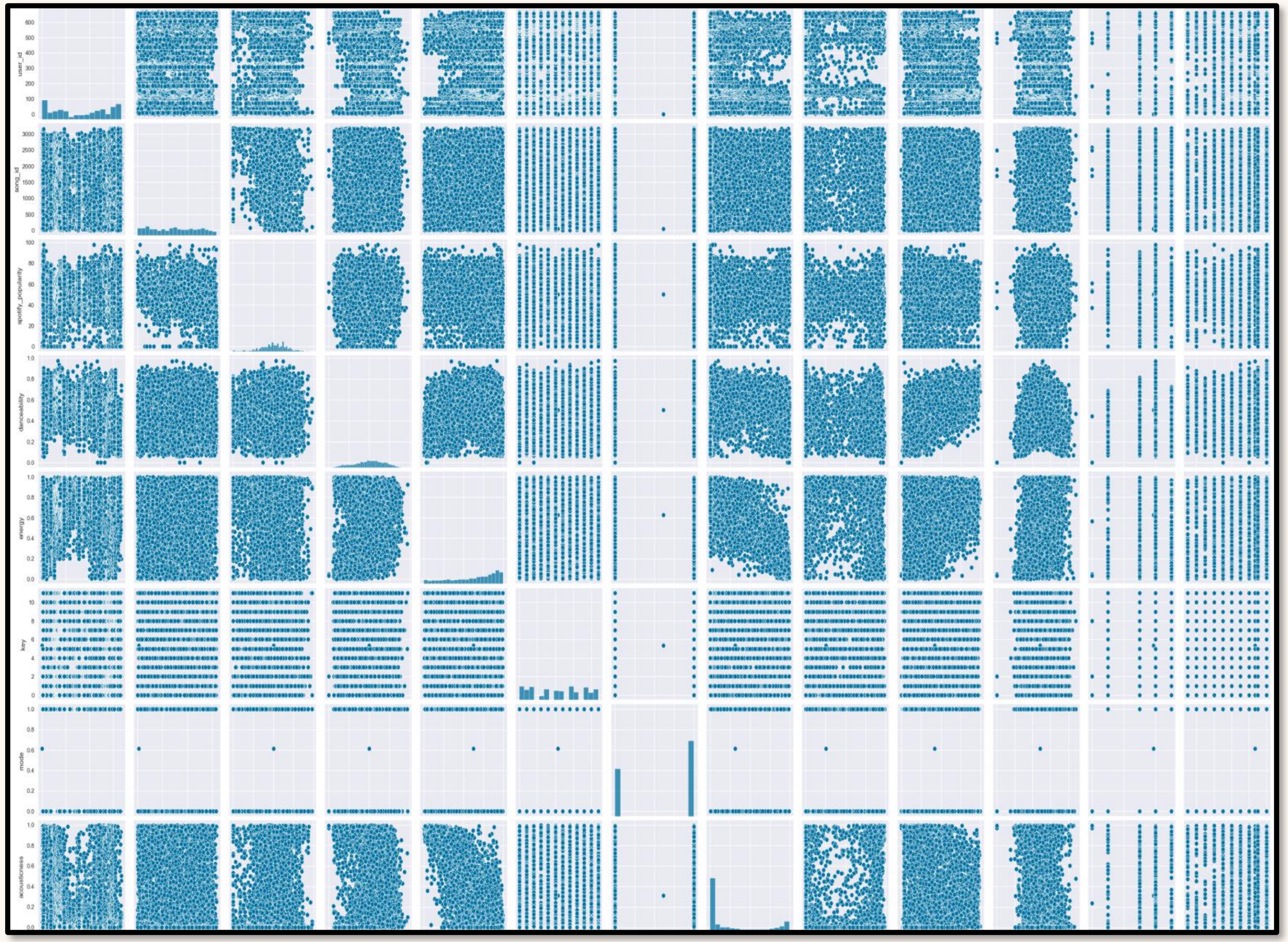
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6998 entries, 0 to 6998
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   user_id             6998 non-null   float64
 1   song_id             6998 non-null   float64
 2   spotify_popularity  6998 non-null   float64
 3   track_name          6998 non-null   object
 4   danceability        6998 non-null   float64
 5   energy              6998 non-null   float64
 6   key                 6998 non-null   float64
 7   mode                6998 non-null   float64
 8   acousticness        6998 non-null   float64
 9   instrumentalness    6998 non-null   float64
 10  valence             6998 non-null   float64
 11  tempo               6998 non-null   float64
 12  time_signature      6998 non-null   float64
 13  track_genre         6998 non-null   object
 14  Rating              6998 non-null   float64
dtypes: float64(13), object(2)
memory usage: 874.8+ KB
```

Data Analysis using Visualization

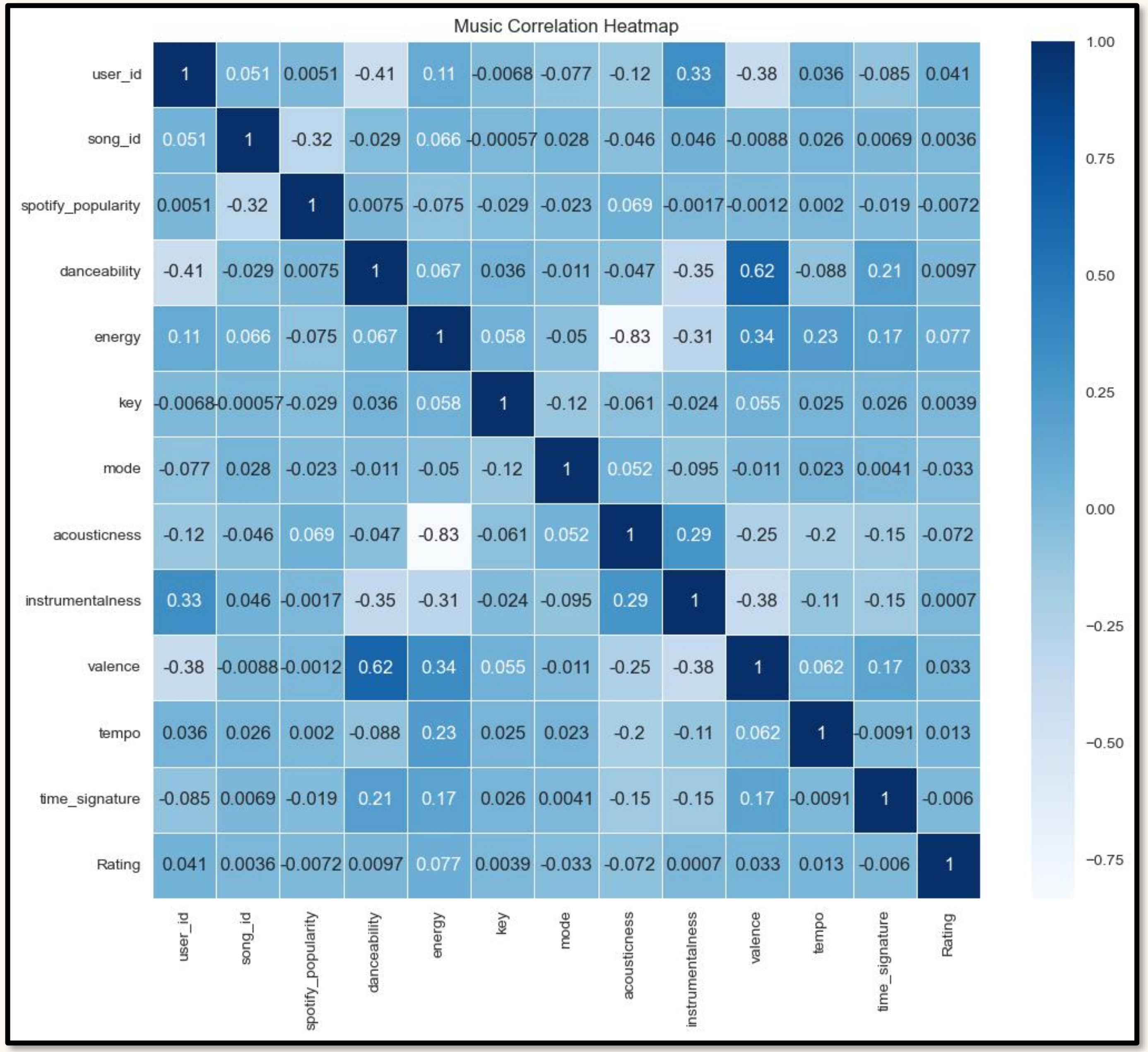
```
correlation_matrix
```

	user_id	song_id	spotify_popularity	danceability	energy	key	mode	acousticness	instrumentalness	valence	tempo	ti
user_id	1.000000	0.051205	0.005080	-0.412936	0.107560	-0.006787	-0.077486	-0.121034	0.333694	-0.384547	0.036491	
song_id	0.051205	1.000000	-0.323414	-0.029409	0.066058	-0.000567	0.027696	-0.046304	0.046352	-0.008793	0.025978	
spotify_popularity	0.005080	-0.323414	1.000000	0.007512	-0.074719	-0.029264	-0.022788	0.068820	-0.001742	-0.001165	0.001989	
danceability	-0.412936	-0.029409	0.007512	1.000000	0.067403	0.035512	-0.010884	-0.046754	-0.350294	0.620873	-0.087664	
energy	0.107560	0.066058	-0.074719	0.067403	1.000000	0.058314	-0.050337	-0.833290	-0.313850	0.341520	0.233372	
key	-0.006787	-0.000567	-0.029264	0.035512	0.058314	1.000000	-0.122768	-0.061337	-0.023520	0.055205	0.025447	
mode	-0.077486	0.027696	-0.022788	-0.010884	-0.050337	-0.122768	1.000000	0.052222	-0.095254	-0.011189	0.023455	
acousticness	-0.121034	-0.046304	0.068820	-0.046754	-0.833290	-0.061337	0.052222	1.000000	0.290770	-0.245238	-0.204798	
instrumentalness	0.333694	0.046352	-0.001742	-0.350294	-0.313850	-0.023520	-0.095254	0.290770	1.000000	-0.381230	-0.105051	
valence	-0.384547	-0.008793	-0.001165	0.620873	0.341520	0.055205	-0.011189	-0.245238	-0.381230	1.000000	0.061609	
tempo	0.036491	0.025978	0.001989	-0.087664	0.233372	0.025447	0.023455	-0.204798	-0.105051	0.061609	1.000000	
time_signature	-0.084930	0.006889	-0.018834	0.208545	0.166912	0.025881	0.004076	-0.149138	-0.151672	0.168946	-0.009101	
Rating	0.040744	0.003629	-0.007232	0.009674	0.077249	0.003900	-0.032774	-0.071977	0.000704	0.032953	0.012882	

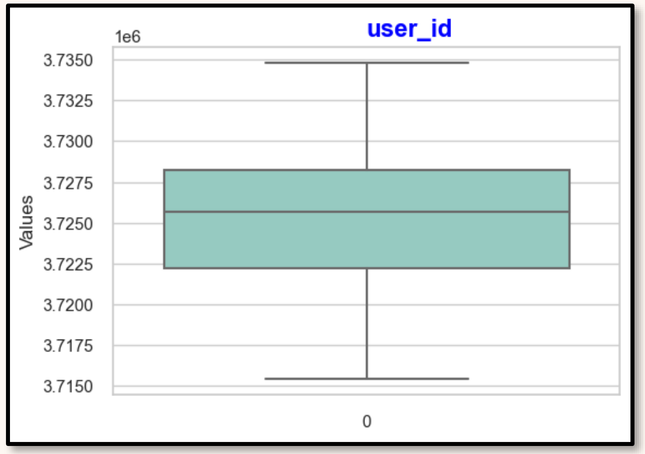
Pairplot for plotting pairwise relationships between variables within a dataset

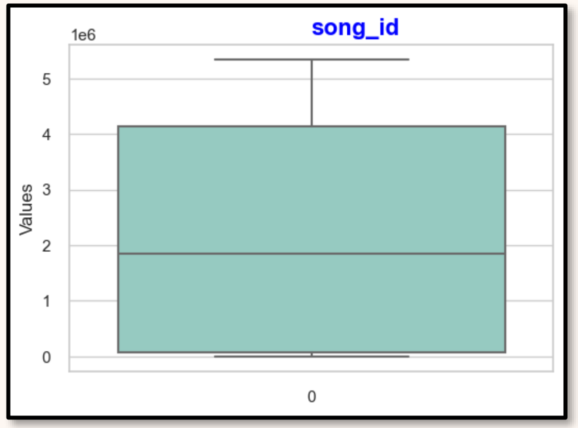


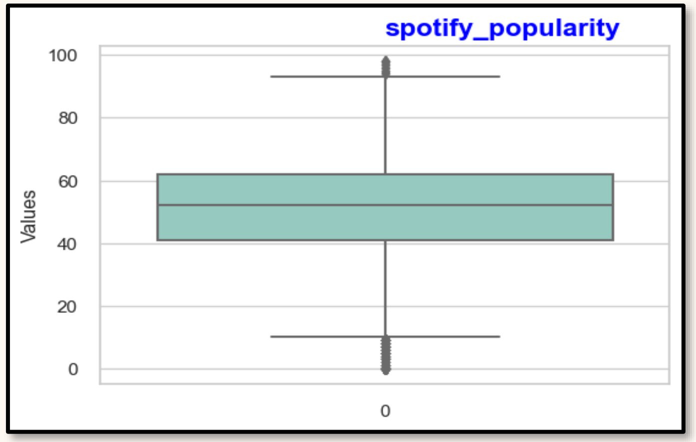
Correlation Heatmap

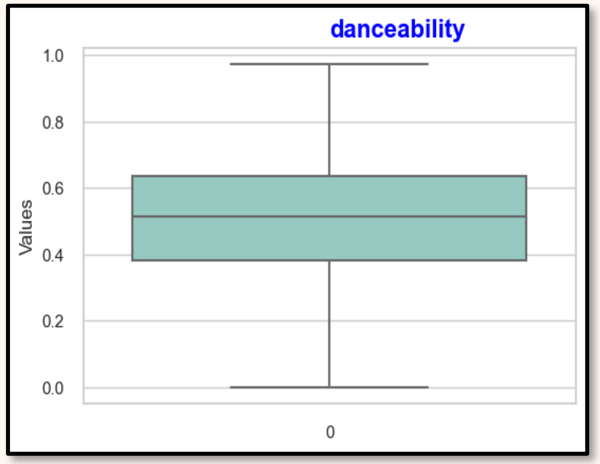


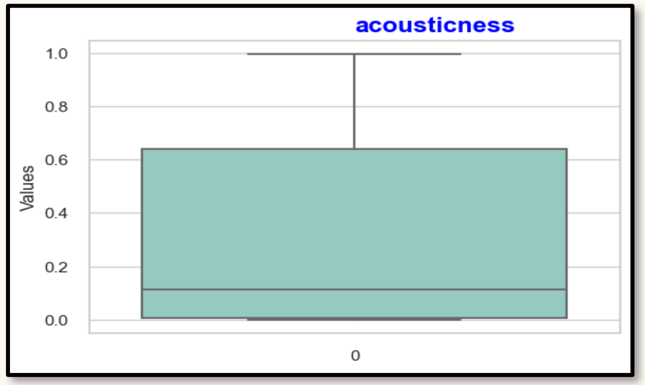
Boxplot for outlier detection

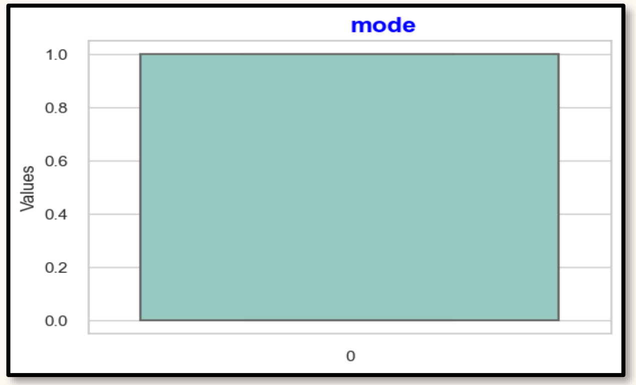


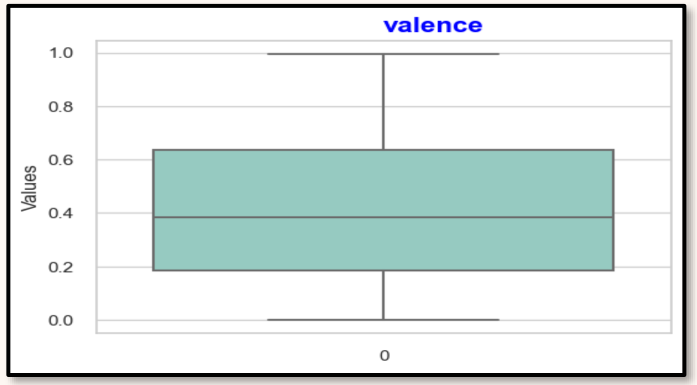


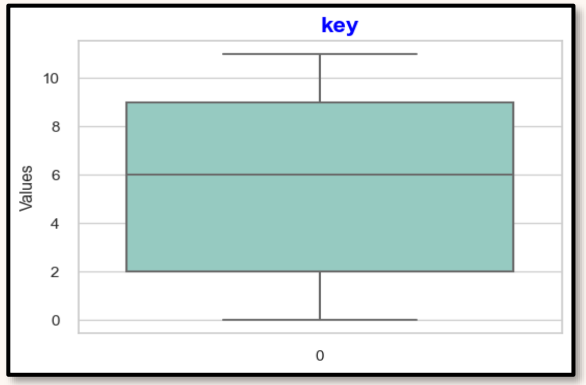


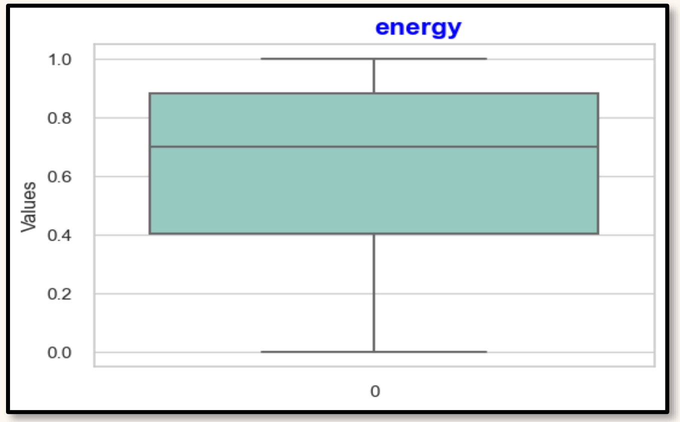


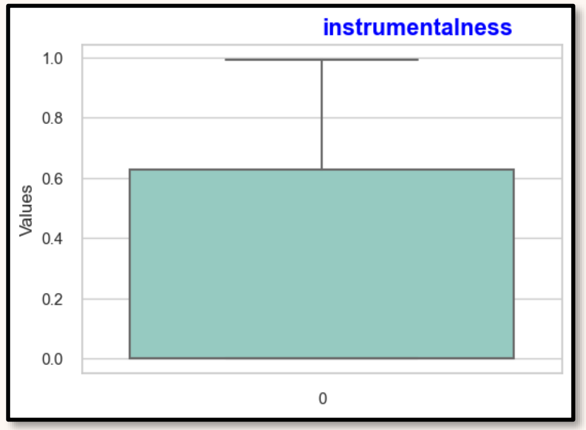


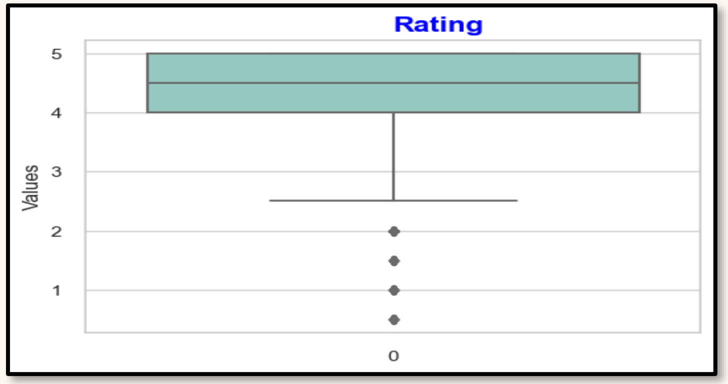


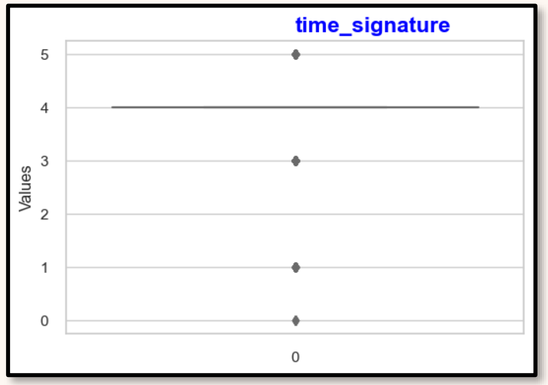




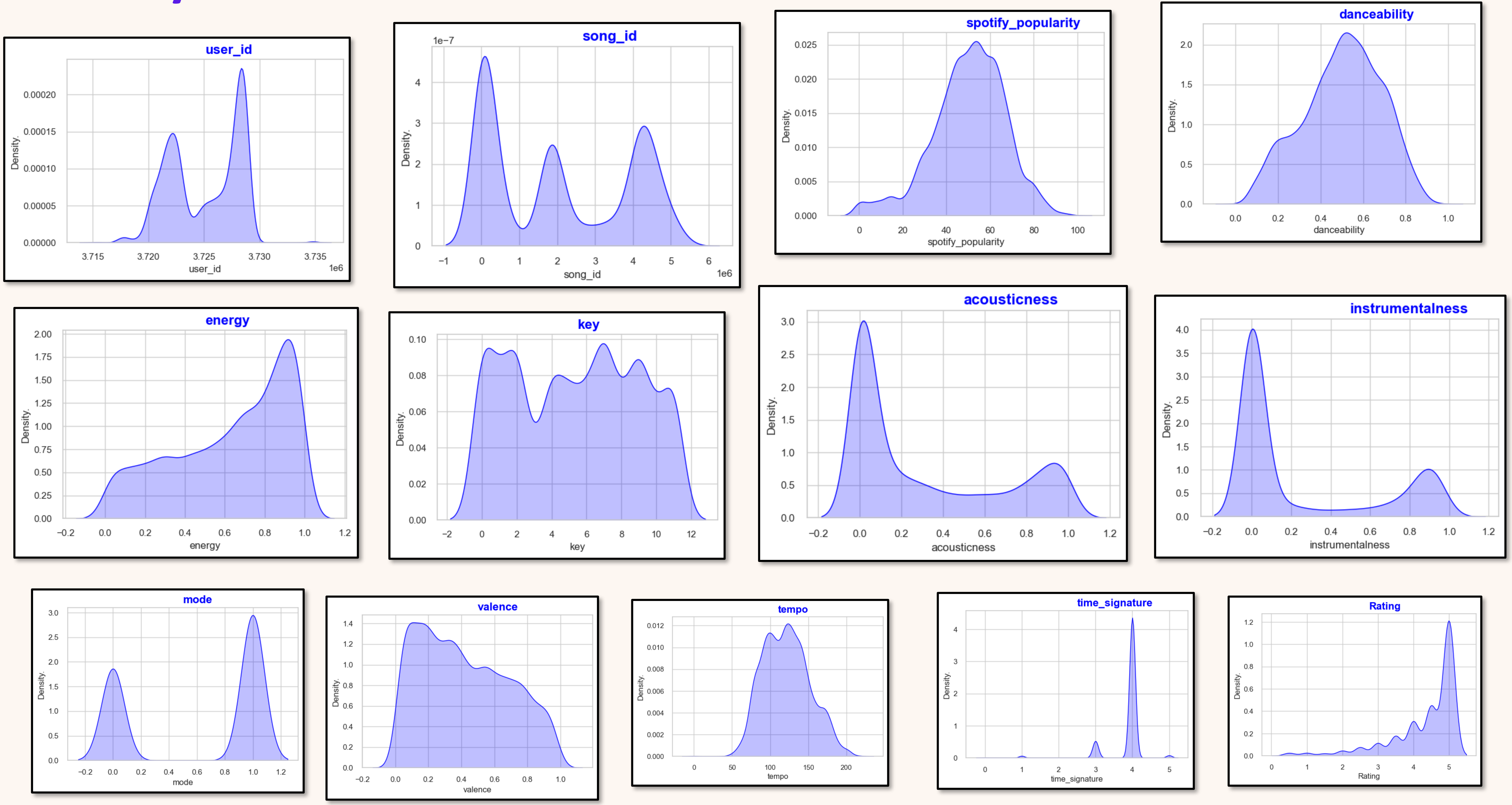




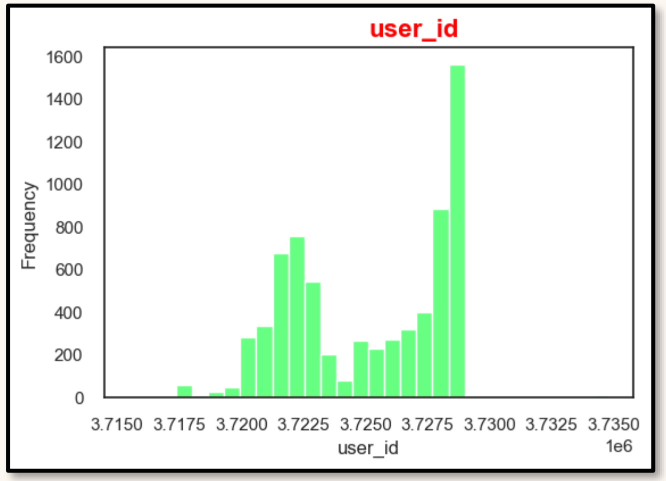


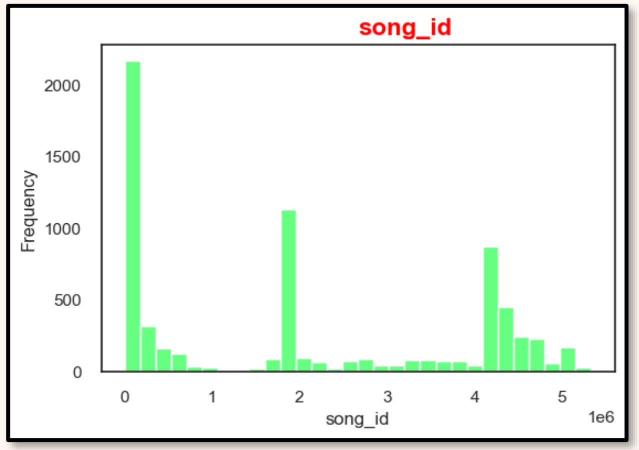


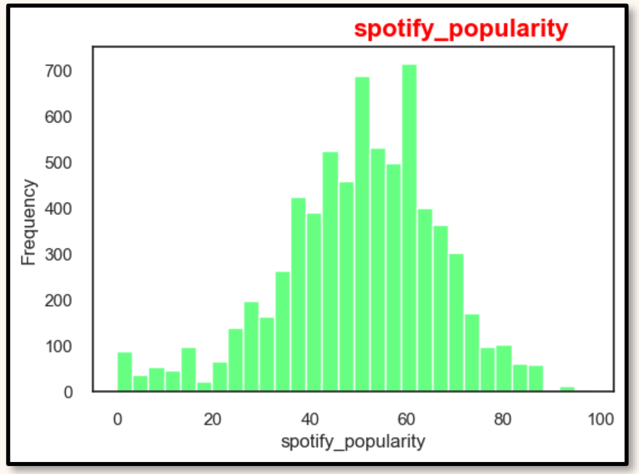
Density Visualization

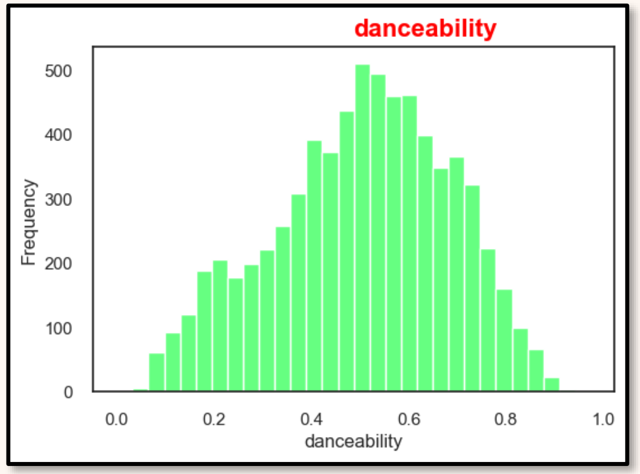


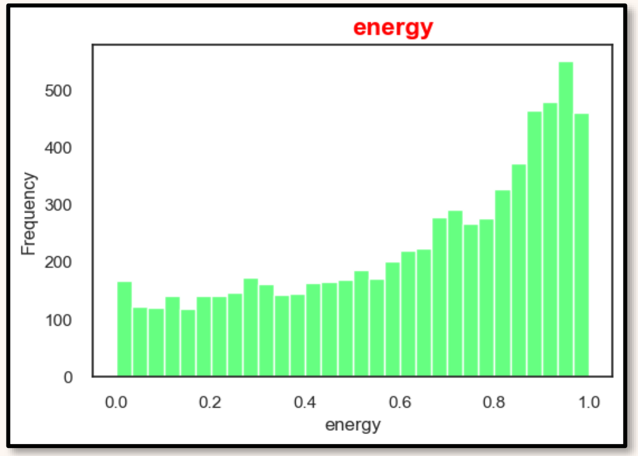
Histogram Visualization

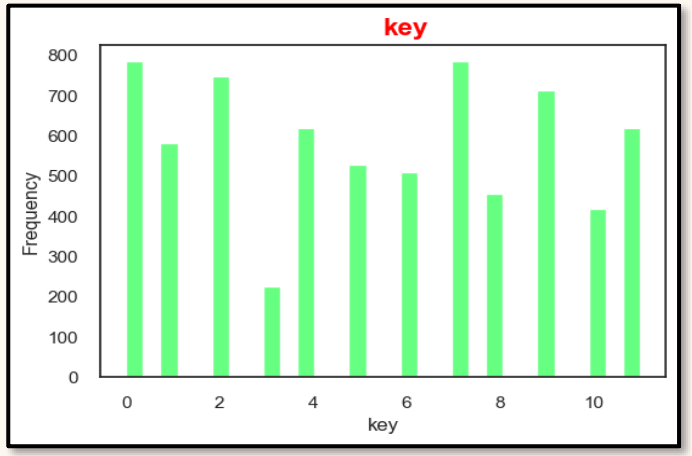


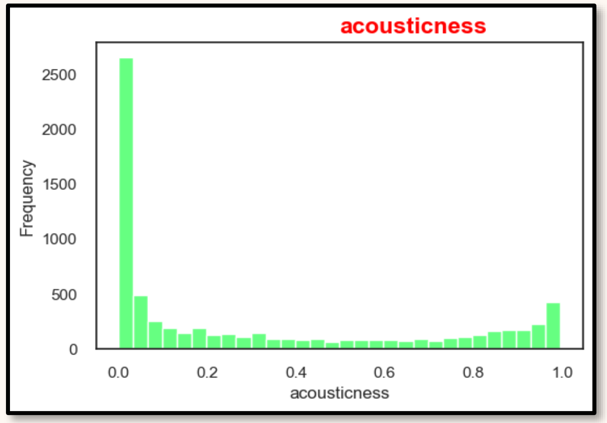


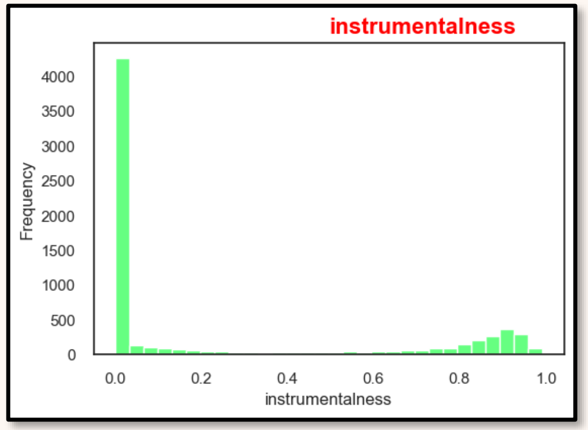


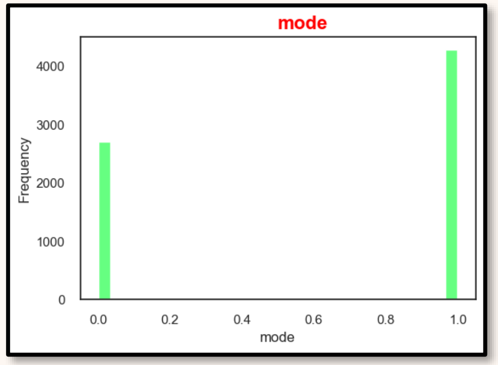


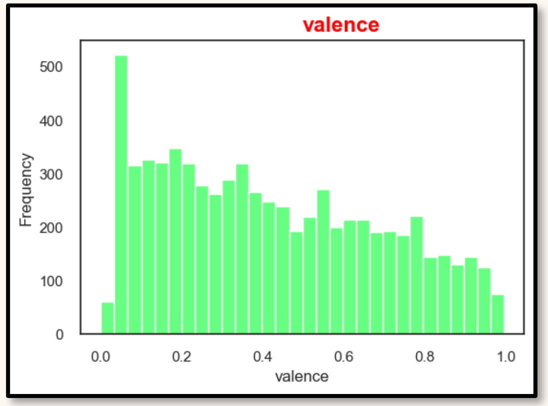


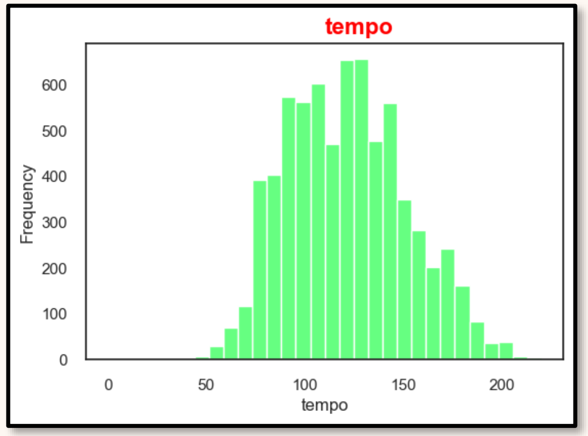


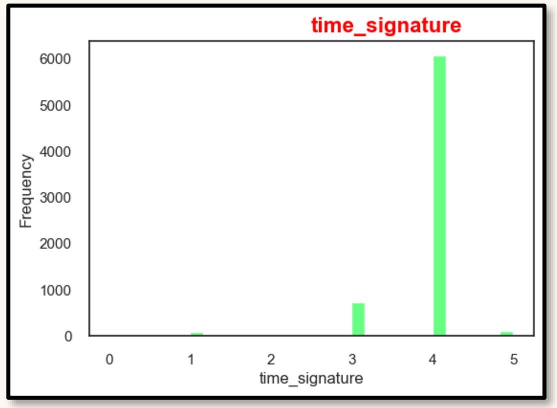


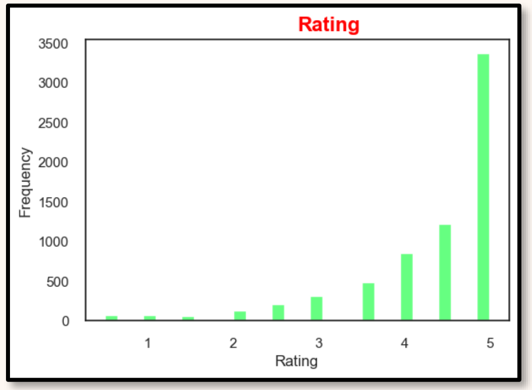












Model Building with Recommendation algorithms :

Collaborative Filtering

By analyzing the listening patterns and preferences of of similar users, the system system recommends songs songs and artists that align align with your taste.

Content-Based Filtering Filtering

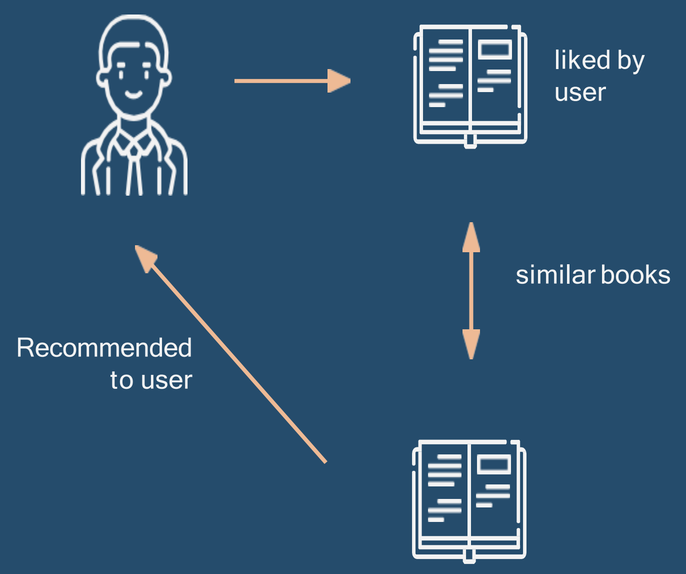
Based on the attributes of songs and artists you enjoy, enjoy, the system suggests suggests similar tracks and and musicians that match your musical preferences.

Hybrid Approaches

Combining collaborative filtering and content-based based filtering techniques, techniques, hybrid algorithms algorithms offer comprehensive and diverse diverse music recommendations.

TYPES OF RECOMMENDATION SYSTEM

Content Based Filtering

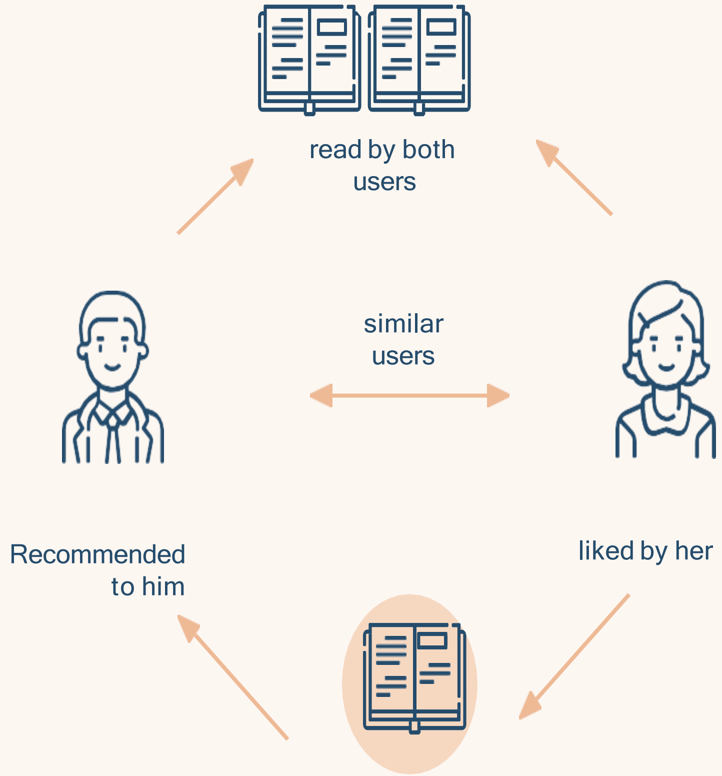


Measure similarity between books

Example: cosine similarity

Process: Text Vector using TfidfVectorizer

Collaborative Filtering



Memory-based:

Predict ratings by learning user’s pattern of giving ratings. Example: KNN

Model-based:

Predict ratings by learning user latent factor and item latent factor. Example: SVD, SVD++

Content Based Filtering

- Content-based filtering uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.
- This filtering makes recommendations by using keywords and attributes assigned to objects in a database and matching them to a user profile.
- Content-based filtering uses similarities in products, services, or content features, as well as information accumulated about the user to make recommendations.
- Content-Based recommender system tries to guess the features or behavior of a user given the item’s features, he/she reacts positively to.

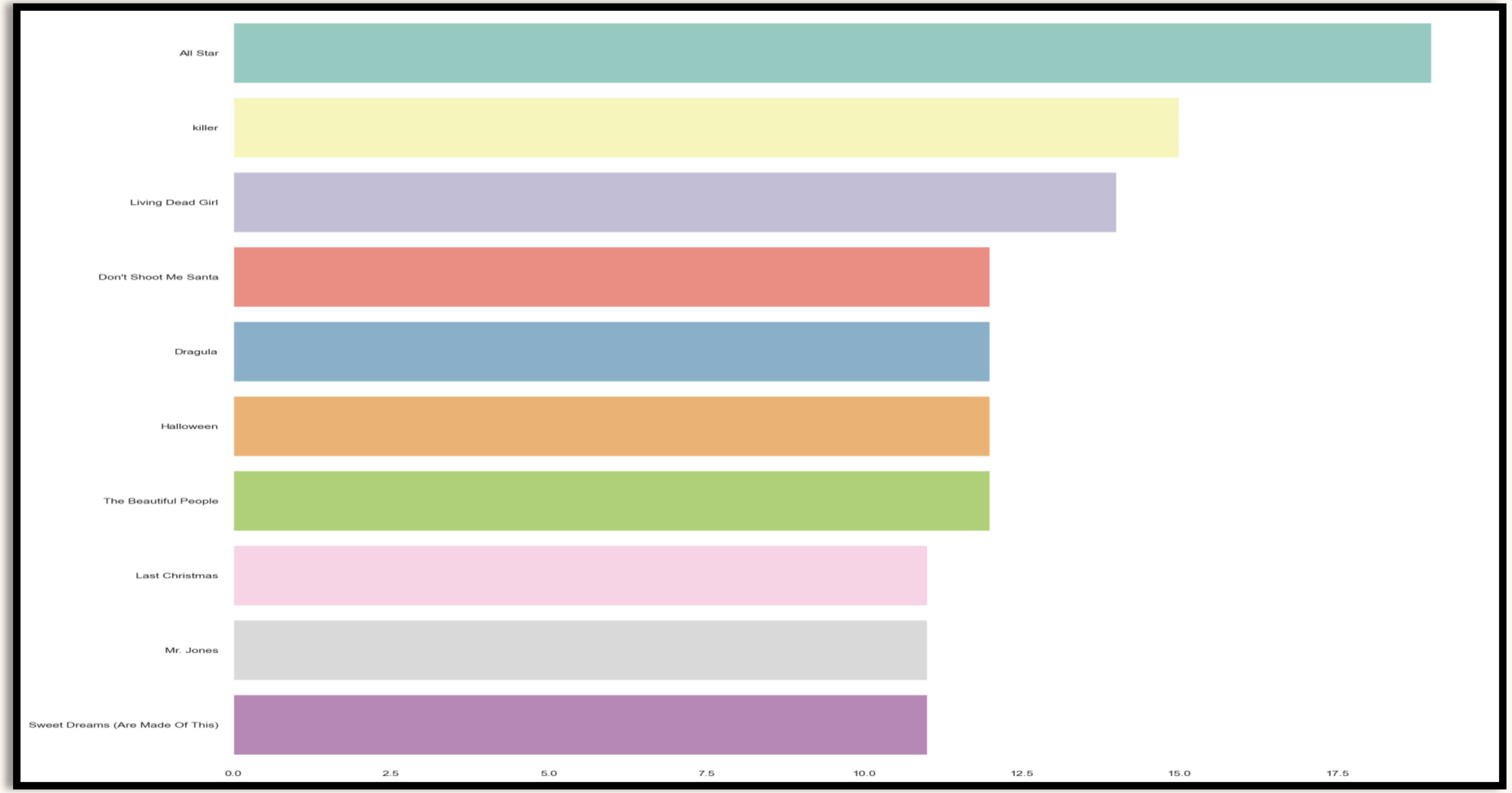
Collaborative Filtering

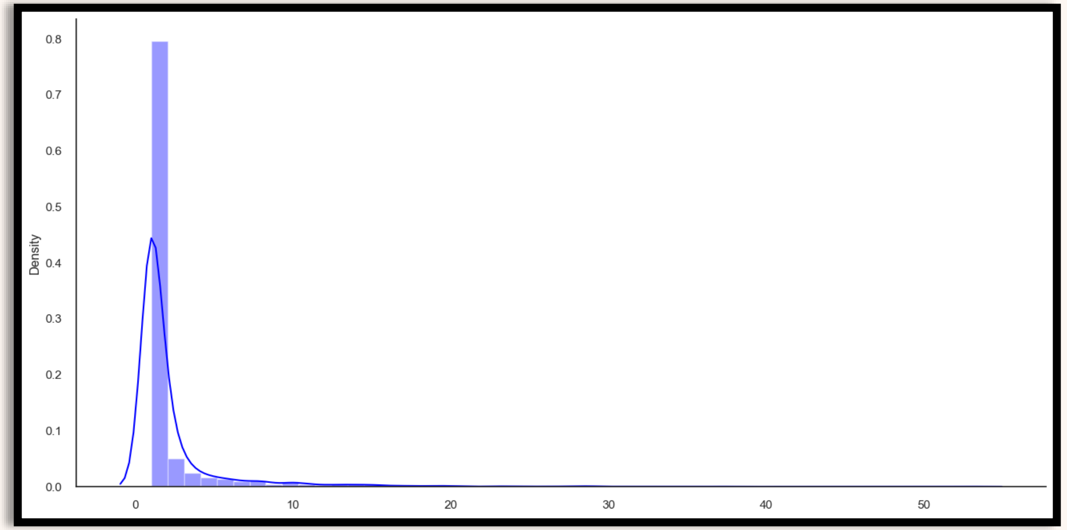
- Collaborative filtering method can filter out items that a user might like on the basis of reactions by similar users.
- This method utilizes preferences and behaviors of other users to come up with recommendations.
- The general operation of these systems is to pair users with similar tastes together into groups. Recommendations are then made based on the collective preferences of the users within each group.
- Thus if user x and user y are both members of the same preference group, and user x likes a particular sample, then there should be a high probability that user y also like this sample. Obviously, this method requires a significantly large initial dataset to provide relevant predictions.

Before moving further, lets explore more about the songs of our data set.

```
top_ten_songs = top_ten_songs[:10]
top_ten_songs
```

	track_name	spotify_popularity	percentage
230	All Star	19	0.27
5345	killer	15	0.21
2648	Living Dead Girl	14	0.20
1234	Don't Shoot Me Santa	12	0.17
1259	Dragula	12	0.17
1864	Halloween	12	0.17
4401	The Beautiful People	12	0.17
2541	Last Christmas	11	0.16
2971	Mr. Jones	11	0.16
4276	Sweet Dreams (Are Made Of This)	11	0.16





A song is listened for an average of 2.203 users, with minimum 1 and maximum 53 users.

Model Building: Model selection and implementation

1

Overview of different models

Compare and evaluate different models models used for music recommendation recommendation systems.

2

Evaluation of model performance performance

Examine the performance metrics and and evaluation methods used to assess assess the effectiveness of our model. model.

3

Implementation of the chosen model

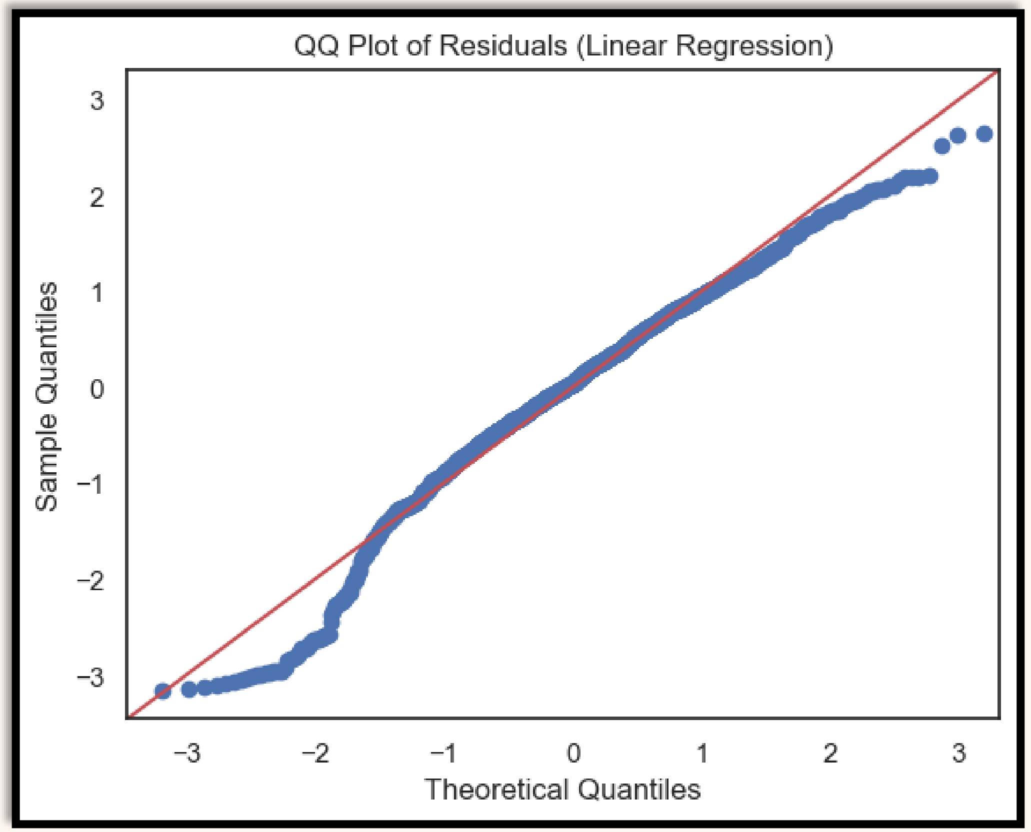
Learn about the step-by-step implementation of our selected model for model for music recommendations.

Overview of different models:

Feature (X) : Danceability, Energy, Key, Mode, Acousticness, Instrumentalness, Valence, Tempo, Time signature

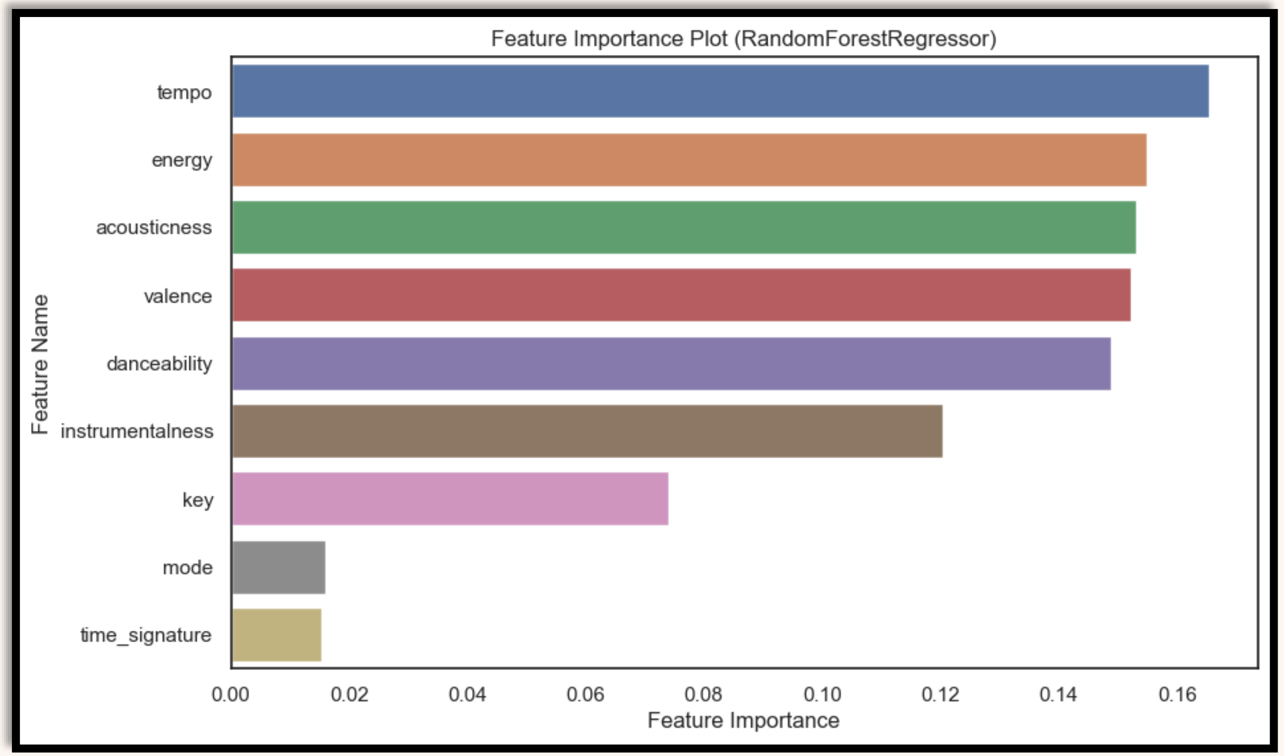
Target (Y) : Spotify Popularity

1. Linear Regression:



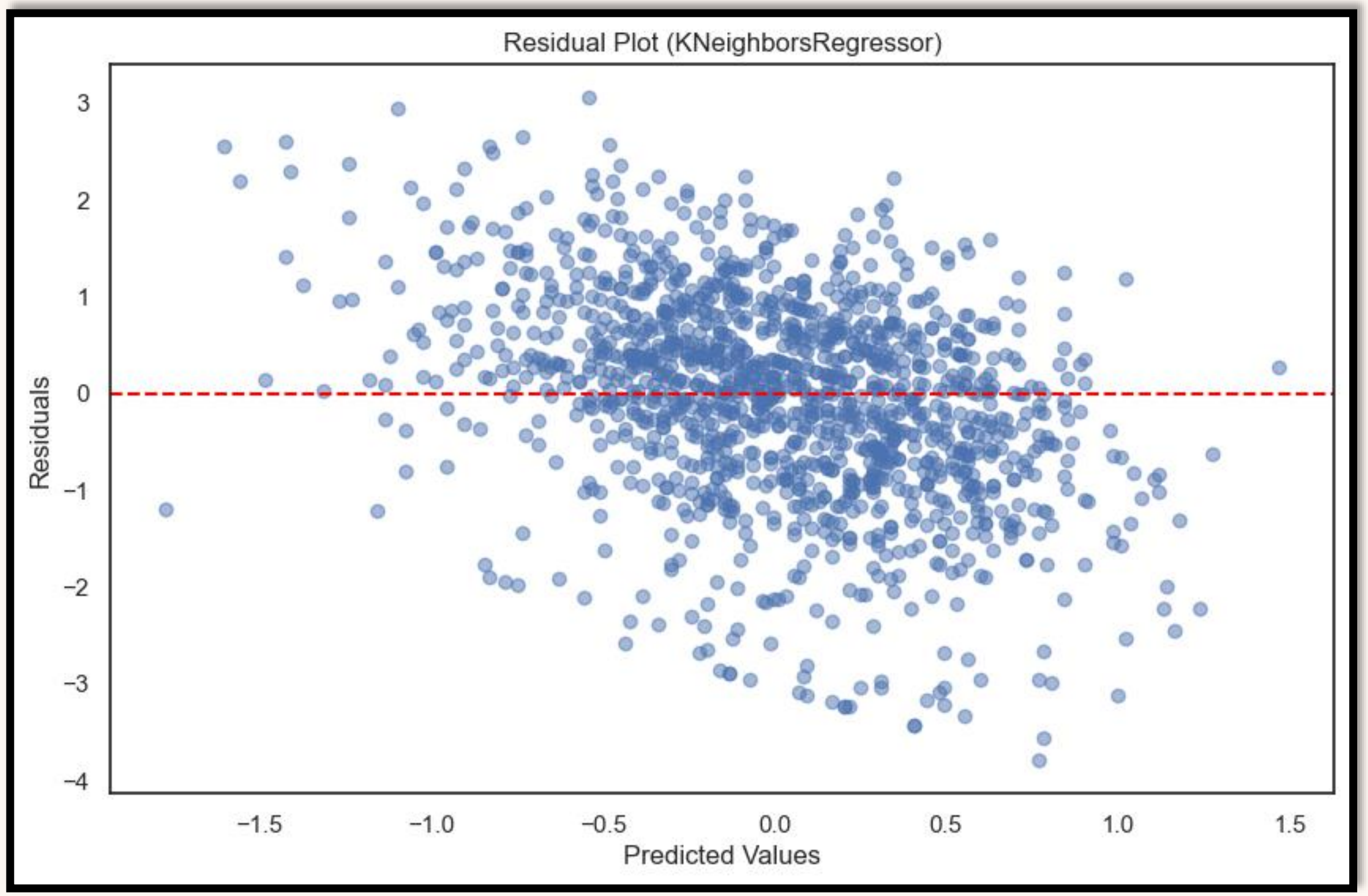

```
Mean Squared Error (MSE): 0.9529041060048986
Mean Absolute Percentage Error (MAPE): 42762.81483899972
```

2. Random Forest Regressor



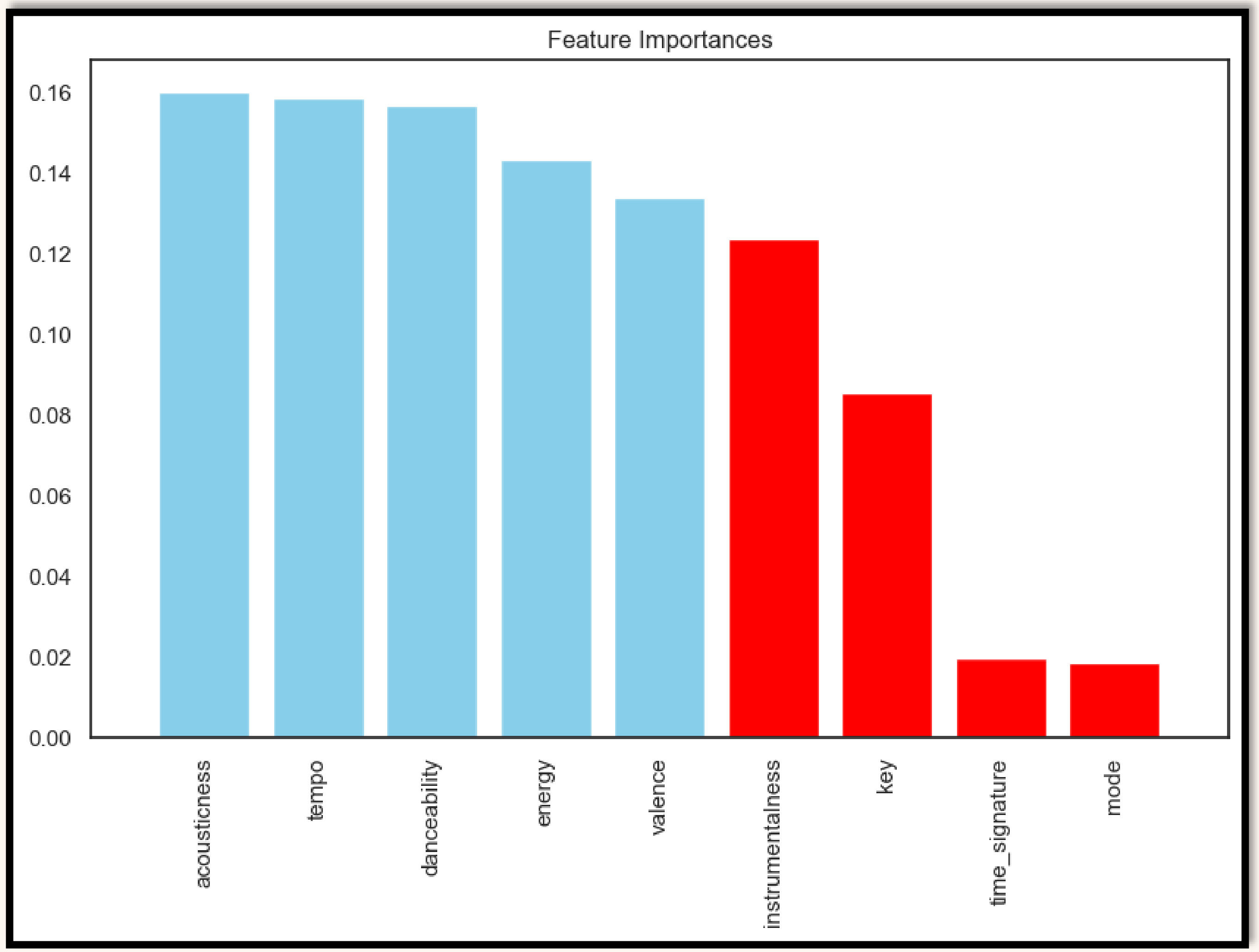

```
Mean Squared Error: 1.061711283186876
Mean Absolute Error: 0.8000961177705082
Mean Absolute Percentage Error: 692100.6050998819
```

3. K Neighbors Regressor



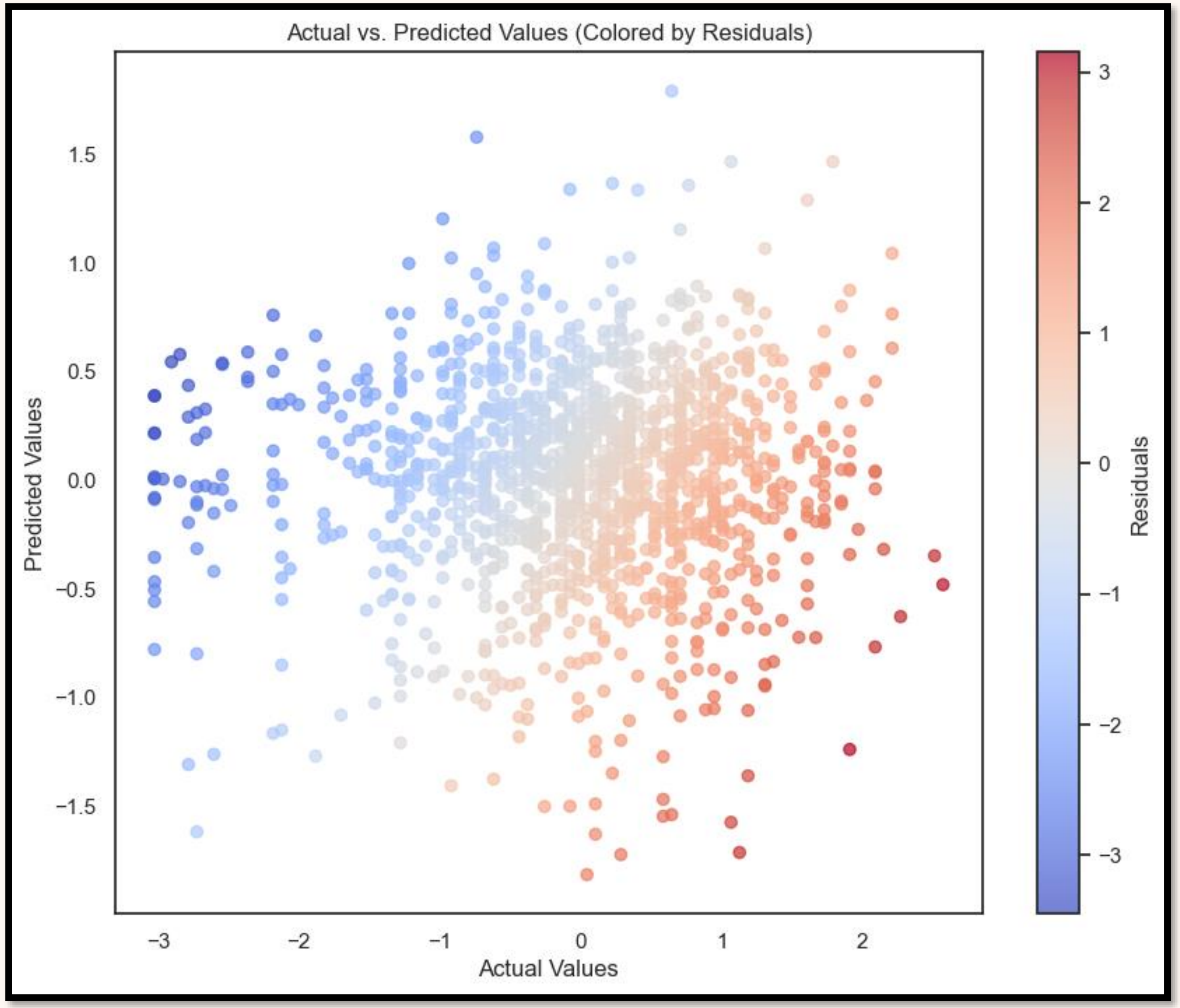

```
Mean Squared Error: 1.1313030223149874
Mean Absolute Error: 0.8239143821498605
Mean Absolute Percentage Error: 9118094.961393878
```

4. Decision Tree Regressor




```
Mean Squared Error: 1.868105736537806
Mean Absolute Error: 1.0662343074105876
Mean Absolute Percentage Error: 11397315.334467398
```

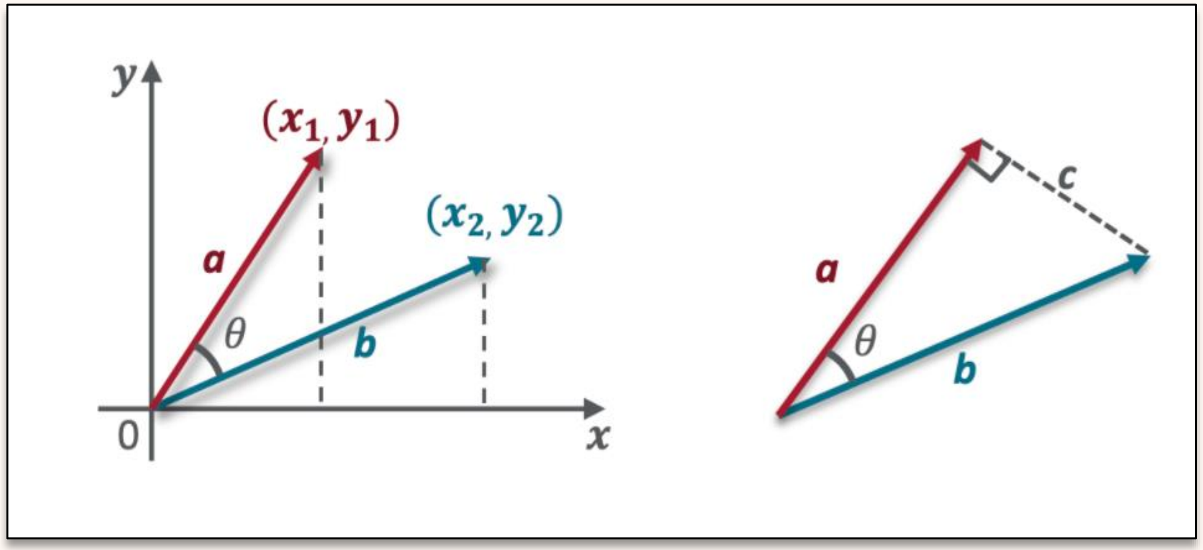
5. XGB Regressor




```
Mean Squared Error: 1.1491712477813893
Mean Absolute Error: 0.8311533711594499
Mean Absolute Percentage Error: 1999235.1559749127
```

6 . Cosine Similarity: Final Model To Recommend

- The metric cosine similarity assesses how similar two or more vectors are.
- The cosine similarity is the cosine of the angle between vectors. In most cases, the vectors are non-zero and belong to the inner product space.
- It is formally described as the difference between the dot product of vectors and the product of the Euclidean norms or magnitude of each vector.



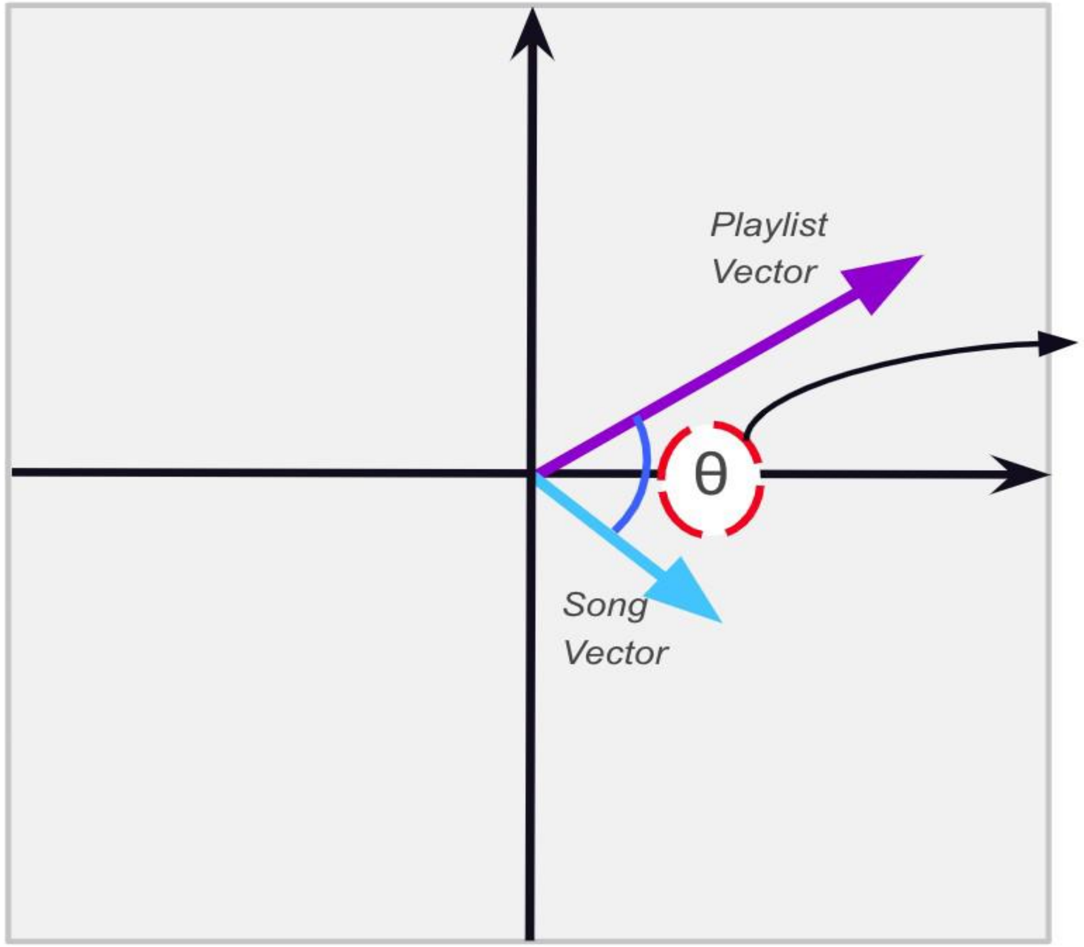
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

The closer the cosine value to 1, the more similar the two vectors are.

Calculating Scores for New Songs

Playlist Vector is compared to individual Song Vectors using *cosine similarity* to generate recommendations:

Cosine Similarity Explained:



This angle represents a personalized score for a new song.
NOTE: Smaller the angle, the higher the song score

Creating User-Item Matrix for User and Songs genre Recommendation

```
user_item_matrix
```

song_id	0	1	2	3	4	5	6	7	8	9	...	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178
user_id																					
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
661	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
662	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
663	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
664	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
665	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

666 rows × 3179 columns

Finding out the similarities between the users by using ' Cosine Similarities Matrix '

```
user_sim_df
```

	0	1	2	3	4	5	6	7	8	9	...	656	657	658	659	660	661	662	663	664
0	1.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
1	0.0	1.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
2	0.0	0.0	1.000000	0.0	0.086789	0.0	0.0	0.000000	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
3	0.0	0.0	0.000000	1.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
4	0.0	0.0	0.086789	0.0	1.000000	0.0	0.0	0.000000	0.0	0.0	...	0.057577	0.041677	0.000000	0.000000	0.036188	0.000000	0.000000	0.000000	0.00000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
661	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.040029	0.0	0.0	...	0.105727	0.051450	0.037050	0.105837	0.119661	1.000000	0.229259	0.176441	0.18650
662	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.084938	0.0	0.0	...	0.307483	0.171946	0.204404	0.286859	0.308078	0.229259	1.000000	0.804424	0.31164
663	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.254388	0.234889	0.152741	0.261302	0.262585	0.176441	0.804424	1.000000	0.47302
664	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.133388	0.137934	0.000000	0.161416	0.222425	0.186501	0.311645	0.473029	1.00000
665	0.0	0.0	0.000000	0.0	0.069271	0.0	0.0	0.000000	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000

666 rows × 666 columns

Cosine value $\propto$ Similarity

Finding out the most similar users.

```
#Most Similar Users

print(user_sim_df.idxmax(axis=1))
print(user_sim_df.max(axis=1).sort_values(ascending=False))
```

```
0        0
1        1
2        2
3        3
4        4
      ...
661    661
662    662
663    663
664    664
665    665
Length: 666, dtype: int64
618    1.0
114    1.0
12     1.0
383    1.0
113    1.0
      ...
409    1.0
517    1.0
24     1.0
97     1.0
492    1.0
Length: 666, dtype: float64
```

We have used the most common recommendation algorithm i.e. "user-user" algorithm because it recommends an item to a user if similar users liked this item before.
The similarity between two users is computed from the amount of items they have in common in the dataset

Encoding all categorical data (features) to numbers(tags) before fitting and evaluation of the model.

```
1 df['tags']
```

```
0       acoustic 0.676 0.461 1.0 -6.746 0.143 0.0322 1...
1       acoustic 0.42 0.166 1.0 -17.235 0.0763 0.924 5...
2       acoustic 0.438 0.359 0.0 -9.734 0.0557 0.21 0....
3       acoustic 0.266 0.0596 0.0 -18.515 0.0363 0.905...
4       acoustic 0.618 0.443 2.0 -9.681 0.0526 0.469 0...
                              ...
6997    black-metal 0.203 0.928 9.0 -10.858 0.0866 2.2...
6998    black-metal 0.193 0.99 1.0 -6.199 0.124 1.19e-...
6999    black-metal 0.573 0.976 1.0 -4.004 0.179 4.57e...
7000    black-metal 0.128 0.954 5.0 -4.753 0.0567 3.53...
7001    black-metal 0.565 0.852 0.0 -3.869 0.034 0.001...
Name: tags, Length: 6794, dtype: object
```

Assigning track name to their respective tags

	track_name	tags
0	comedy	acoustic 0.676 0.461 1.0 -6.746 0.143 0.0322 1...
1	ghost - acoustic	acoustic 0.42 0.166 1.0 -17.235 0.0763 0.924 5...
2	to begin again	acoustic 0.438 0.359 0.0 -9.734 0.0557 0.21 0....
3	can't help falling in love	acoustic 0.266 0.0596 0.0 -18.515 0.0363 0.905...
4	hold on	acoustic 0.618 0.443 2.0 -9.681 0.0526 0.469 0...
...	...	...
6997	stand tall in fire	black-metal 0.203 0.928 9.0 -10.858 0.0866 2.2...
6998	antichrist siege machine	black-metal 0.193 0.99 1.0 -6.199 0.124 1.19e-...
6999	the scope of obsession	black-metal 0.573 0.976 1.0 -4.004 0.179 4.57e...
7000	phantoms of mortem tales	black-metal 0.128 0.954 5.0 -4.753 0.0567 3.53...
7001	bergagasten	black-metal 0.565 0.852 0.0 -3.869 0.034 0.001...

6794 rows × 2 columns

The final result:

Recommended songs for Users

```python
user_id_to_recommend = 117
track_genre_to_recommend = 'acoustic'

recommended_tracks = recommend_songs(user_id_to_recommend, user_item_matrix, user_similarity, track_genre_to_recommend, Musi

print("Recommended songs for User {} in the {} genre:".format(user_id_to_recommend, track_genre_to_recommend))
print('=================================================================================================================')
for i, (track_name, rating) in enumerate(recommended_tracks, start=1):
    print("{}. {}, Rating:  {}".format(i, track_name, rating))
```

```
Recommended songs for User 117 in the acoustic genre:
=================================================================================================================
1. Mujer con Abanico, Rating:  5.0
2. Fallen Star, Rating:  5.0
3. Bouncing Bona, Rating:  5.0
4. そういえば今日から化け物になった, Rating:  5.0
5. 2002 - Acoustic, Rating:  5.0
6. death bed (coffee for your head), Rating:  5.0
7. My Baby's Cheating - I Sure Got the Feeling, Rating:  5.0
8. death bed (coffee for your head), Rating:  5.0
9. Disco Ball, Rating:  5.0
10. Ghost - Acoustic, Rating:  5.0
```

Deploying the system

Selecting an appropriate deployment platform

Choose the ideal platform to to deploy our music recommendation system for for maximum accessibility. accessibility.

Scaling the system for large user bases

Find out how we ensured the the scalability of our system system to cater to a growing growing user base.

Ensuring system stability stability and reliability reliability

Learn about the measures taken to maintain the stability stability and reliability of the the recommendation system. system.

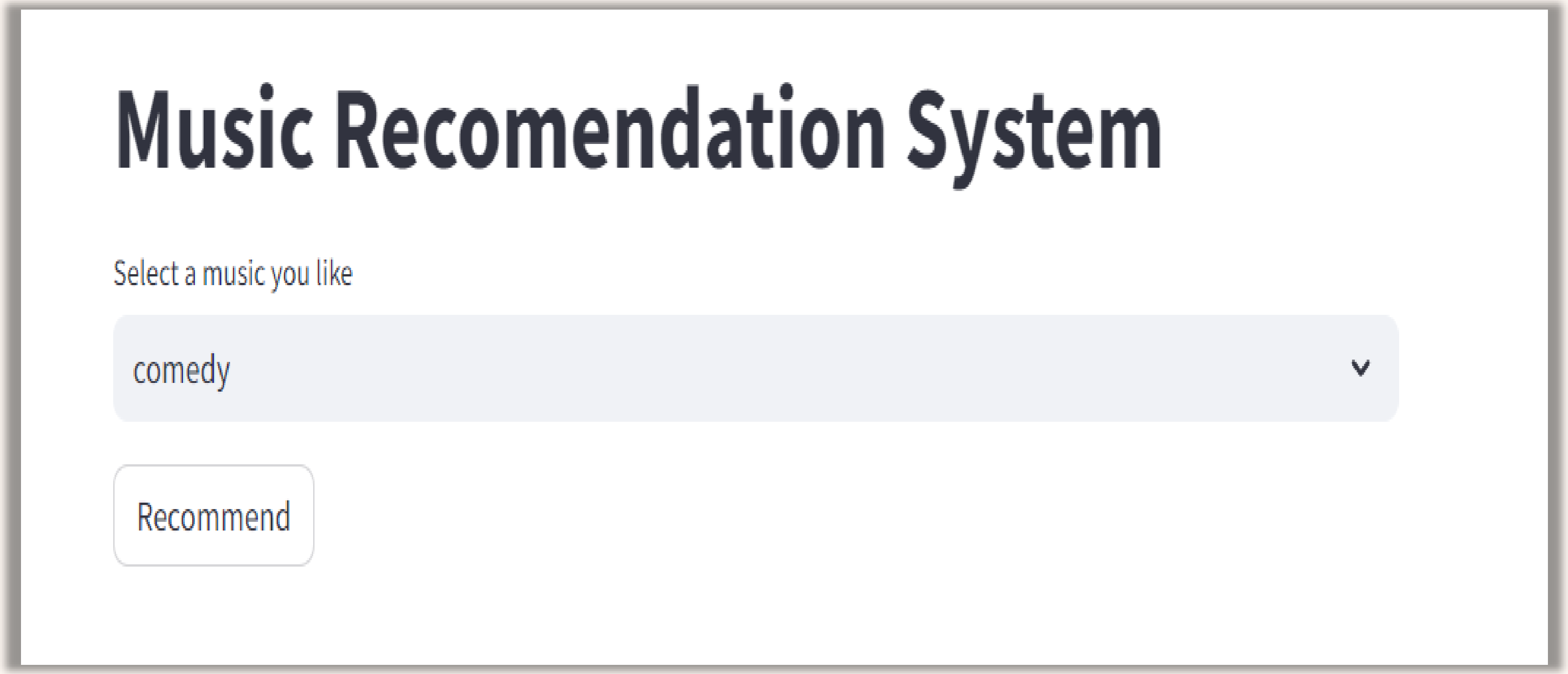
Importing Streamlit and Pickle

```
import streamlit as st
import pickle
```

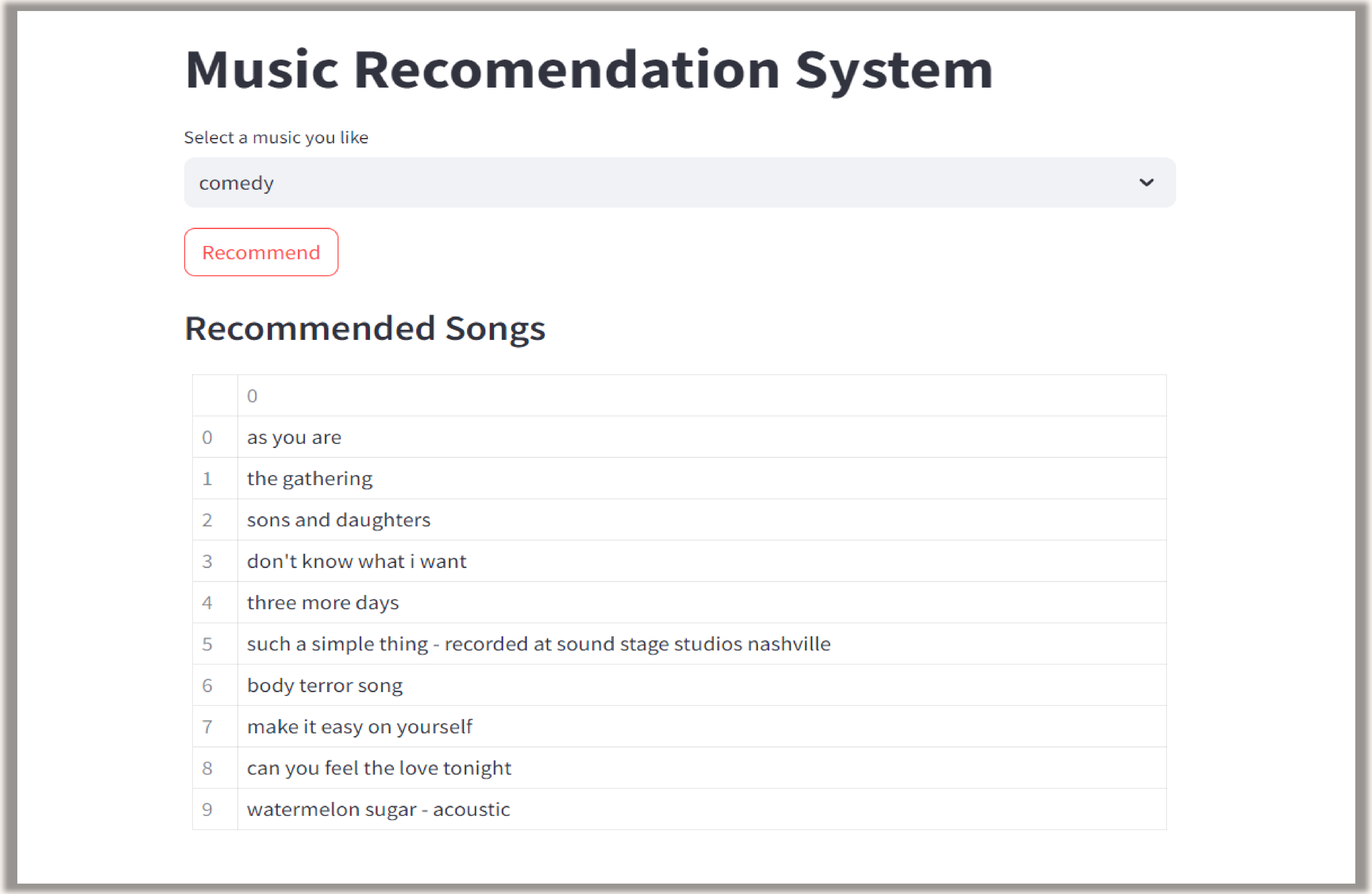
```
selected_music_name = st.selectbox('Select a music you like', music['title'].values)

if st.button('Recommend'):
    names = recommend(selected_music_name)
    st.subheader('Recommended Songs')
    st.table(names[0])
```

Running Streamlit

```
PS D:\Project\Recommendation_engine_Project> streamlit run Deployment.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.180.146:8501
```



Click Recommend to Recommend the songs Similar to the selected song


Music Recomendation System
Select a music you like
comedy
Recommend
Recommended Songs
0
0 as you are
1 the gathering
2 sons and daughters
3 don't know what i want
4 three more days
5 such a simple thing - recorded at sound stage studios nashville
6 body terror song
7 make it easy on yourself
8 can you feel the love tonight
9 watermelon sugar - acoustic

Conclusion

1 Recap of the project

This assignment provided us with a fantastic learning opportunity. We've studied data mining, data cleaning, visualization, Model building and a lot more while working on this project.

2 Impact of the music recommendation system system

This recommendation engine make your platform maximum personalized. It automate curating and playlisting audio.

This system provide quality and immersive customer streaming experience and also get insights about users' behavior and make data-based marketing decisions.

3 Future enhancements and possibilities

We were unable to create a model utilizing singular value decomposition and support vector machines due to a lack of time. Also, there is a lot of potential in combining several dimensions in this music recommender systems. We will try to work on the same in the near future.


THANK YOU