Music Recommendation System with Natural Language Processing

Introduction.

We designed a music recommendation system using a dataset of over 28,000 Spotify tracks containing both audio features and full lyrics. Our goal was to build two systems: one based on traditional content-based filtering, and another enhanced by Natural Language Processing (NLP) to capture lyrical sentiment.

For our project, we chose to use a Spotify data set from Kaggle and create two recommender systems. One of these recommender systems used simpler content-based filtering while the other was based around our own sentiment analysis of the song lyrics. Our data set had just over 28,000 observations and 31 total variables, including 'energy', 'acousticness', and other measures of song characteristics that we used to generate similarity scores. We chose this data set for our project because we are all passionate about music, and we were curious to see if a recommender system like this could help us to discover new music.

Data Description.

The chosen data set for our project is a data set of songs released from 1950-2019, from Kaggle (\url{https://www.kaggle.com/datasets/saurabhshahane/music-dataset-1950-to-2019}). The data has 28,372 total observations and 31 columns/variables. Here are all of them listed:

- Unnamed: 0
- artist_name
- track_name
- release_date
- genre
- lyrics
- len
- dating

- violence
- world/life
- night/time
- shake the audience
- family/gospel
- romantic
- communication
- obscene
- music
- movement/places
- light/visual perceptions
- family/spiritual
- like/girls
- sadness
- feelings
- danceability
- loudness
- acousticness
- instrumentalness
- valence
- energy
- topic
- age.

All of the variables in this data set will be predictors in some way, given that we will be creating a recommendation system based on all inputs, although we will focus more specifically on ‘lyrics’ for creating our sentiment analysis output. Many of these variables are numeric, but there are also string and longer text (lyrics) outputs as well. It appears that there is no missing data, which means that our cleaning process should be fairly easy and this data set is nearly ready to be analyzed.

Background Information

There are two primary types of recommender systems:

- **Content-based** systems recommend items by analyzing the properties or attributes of the items themselves (such as audio features and lyrics in our case). These systems recommend new items by comparing item feature vectors using similarity metrics such as cosine similarity.

  > *Example:* Songs with similar energy, danceability, and lyrical content are recommended to a user based on previously liked songs.

- **Collaborative filtering** systems, by contrast, rely on user interaction data (such as song ratings or listening history) to find patterns across users or items. These methods compute similarity either between users or between items based on shared behavior.
  - In a **user-based** system, the algorithm compares the listening histories of different users to find similar users.
  - In an **item-based** system, items (e.g., songs) are compared based on how similar users have rated them.

Cosine Similarity

Across both types of systems, the metric of **cosine similarity** is commonly used to measure how similar two vectors are.
In the context of music, these vectors could represent:

- Audio features (e.g., `danceability`, `energy`)
- TF-IDF word embeddings of lyrics
- Combined feature sets including sentiment

Interpretation of values:

- **+1**: Perfect similarity
- **0**: No similarity (orthogonal)
- **-1**: Complete opposition

```
from IPython.display import display, Math

display(Math(r'S_C(A, B) := \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \

```

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

Application in Our Music Recommender System

In our project, we developed two content-based recommender systems using Spotify song metadata and lyrics:

1. **Basic Recommender**
   Uses standardized numeric audio features like `danceability`, `energy`, `valence`, etc., along with reduced lyric vectors from TF-IDF + Truncated SVD.
   Cosine similarity identifies the 10 most similar songs to the input track.
2. **NLP-Enhanced Recommender**
   Incorporates **sentiment analysis** using TextBlob to score lyrics from -1 to +1.
   These scores are combined with audio and textual features.
   Word clouds were generated to visualize word distributions for positive and negative sentiment songs.

In both systems, **cosine similarity** serves as the backbone for identifying emotional and musical similarity between tracks.

Load and Explore Dataset

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors
from sklearn.decomposition import TruncatedSVD
from wordcloud import WordCloud
```

```
# Ensure necessary NLTK resources are downloaded
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
file_path = "/content/tcc_ceds_music.csv"  # Path to the uploaded file in C
df = pd.read_csv(file_path)
```

```
# Display first few rows and dataset info
display(df.head())
display(df.info())
```

	Unnamed: 0	artist_name	track_name	release_date	genre	lyrics	len	da
0	0	mukesh	mohabbat bhi jhoothi	1950	pop	hold time feel break feel untrue convince spea...	95	0.00
1	4	frankie laine	i believe	1950	pop	believe drop rain fall grow believe darkest ni...	51	0.03
						sweetheart		

2	6	johnnie ray	cry	1950	pop	send letter goodbye secret feel bet...	24	0.00
3	10	pérez prado	patricia	1950	pop	kiss lips want stroll charm mambo chacha merin...	54	0.04
4	12	giorgos papadopoulos	apopse eida oneiro	1950	pop	till darling till matter know till dream live ...	48	0.00

5 rows × 31 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28372 entries, 0 to 28371
Data columns (total 31 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Unnamed: 0                28372 non-null  int64  
 1   artist_name               28372 non-null  object 
 2   track_name                28372 non-null  object 
 3   release_date              28372 non-null  int64  
 4   genre                     28372 non-null  object 
 5   lyrics                    28372 non-null  object 
 6   len                       28372 non-null  int64  
 7   dating                    28372 non-null  float64
 8   violence                  28372 non-null  float64
 9   world/life                28372 non-null  float64
 10  night/time                28372 non-null  float64
 11  shake the audience        28372 non-null  float64
 12  family/gospel             28372 non-null  float64
 13  romantic                  28372 non-null  float64
 14  communication             28372 non-null  float64
 15  obscene                   28372 non-null  float64
 16  music                     28372 non-null  float64
 17  movement/places           28372 non-null  float64
 18  light/visual perceptions  28372 non-null  float64
 19  family/spiritual          28372 non-null  float64
 20  like/girls                28372 non-null  float64
 21  sadness                   28372 non-null  float64
 22  feelings                  28372 non-null  float64
 23  danceability              28372 non-null  float64
 24  loudness                  28372 non-null  float64
 25  acousticness              28372 non-null  float64
 26  instrumentalness          28372 non-null  float64
 27  valence                   28372 non-null  float64
 28  energy                    28372 non-null  float64
 29  topic                     28372 non-null  object 
 30  age                       28372 non-null  float64
```

```
 30  age                         28372 non-null  float64
dtypes: float64(23), int64(3), object(5)
memory usage: 6.7+ MB
None
```

As can be seen, we have 30 columns and 28,372 rows. There are no missing values in any column which makes our analysis easier and we do not have to do any imputation. Most of the variables are float64 data types with the exception being artist_name, track_name, genre, lyrics, and age which are objects. release_date and len are int64 variables which are still numeric. This means we have 25 numeric variables and 5 categorical ones.

EDA.

We will begin by examining the dataset, by viewing the head to get a sense of the column layouts and their values. We will then use the info command to check the data types of each variable as well as the non-null count.

```
# Drop unnecessary columns
df.drop(columns=['Unnamed: 0'], inplace=True)
```

```
# For categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
categorical_cols = categorical_cols[categorical_cols != 'lyrics']
for col in categorical_cols:
    print(f'{col} unique values: {df[col].nunique()}')
    print(df[col].value_counts().head())  # Show top 5 most frequent catego

# Data appears to be mostly numerical types with the majority in float64 ty
# Genre, Lyrics, and Topic.
# For numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
print(df[numerical_cols].describe())
```

```
artist_name unique values: 5426
artist_name
johnny cash        190
ella fitzgerald    188
dean martin        146
willie nelson      131
george jones       107
Name: count, dtype: int64
track_name unique values: 23689
```

```
track_name
tonight        17
stay           15
hold on        15
without you    14
home           13
Name: count, dtype: int64
genre unique values: 7
genre
pop        7042
country    5445
blues      4604
rock       4034
jazz       3845
Name: count, dtype: int64
topic unique values: 8
topic
sadness       6096
violence      5710
world/life    5420
obscene       4882
music         2303
Name: count, dtype: int64
       release_date           len        dating      violence    world/life
count  28372.000000  28372.000000  28372.000000  28372.000000  28372.000000
mean    1990.236888     73.028444      0.021112      0.118396      0.120973
std       18.487463     41.829831      0.052370      0.178684      0.172200
min     1950.000000      1.000000      0.000291      0.000284      0.000291
25%     1975.000000     42.000000      0.000923      0.001120      0.001170
50%     1991.000000     63.000000      0.001462      0.002506      0.006579
75%     2007.000000     93.000000      0.004049      0.192608      0.197793
max     2019.000000    199.000000      0.647706      0.981781      0.962105

         night/time  shake the audience  family/gospel      romantic  \
count  28372.000000        28372.000000   28372.000000  28372.000000
mean       0.057387            0.017422       0.017045      0.048681
std        0.111923            0.040670       0.041966      0.106095
min        0.000289            0.000284       0.000289      0.000284
25%        0.001032            0.000993       0.000923      0.000975
50%        0.001949            0.001595       0.001504      0.001754
75%        0.065842            0.010002       0.004785      0.042301
max        0.973684            0.497463       0.545303      0.940789

       communication  ...    like/girls       sadness      feelings  \
count   28372.000000  ...  28372.000000  28372.000000  28372.000000
mean        0.076680  ...      0.028057      0.129389      0.030996
std         0.109538  ...      0.058473      0.181143      0.071652
min         0.000291  ...      0.000284      0.000284      0.000289
25%         0.001144  ...      0.000975      0.001144      0.000993
50%         0.002632           0.001595      0.005263      0.001754
```

We will now look at the Top 5 most common values for each categorical variable. We can see Johnny Cash is the most occuring artist, sadness is the most popular topic, and pop is the most popular genre.

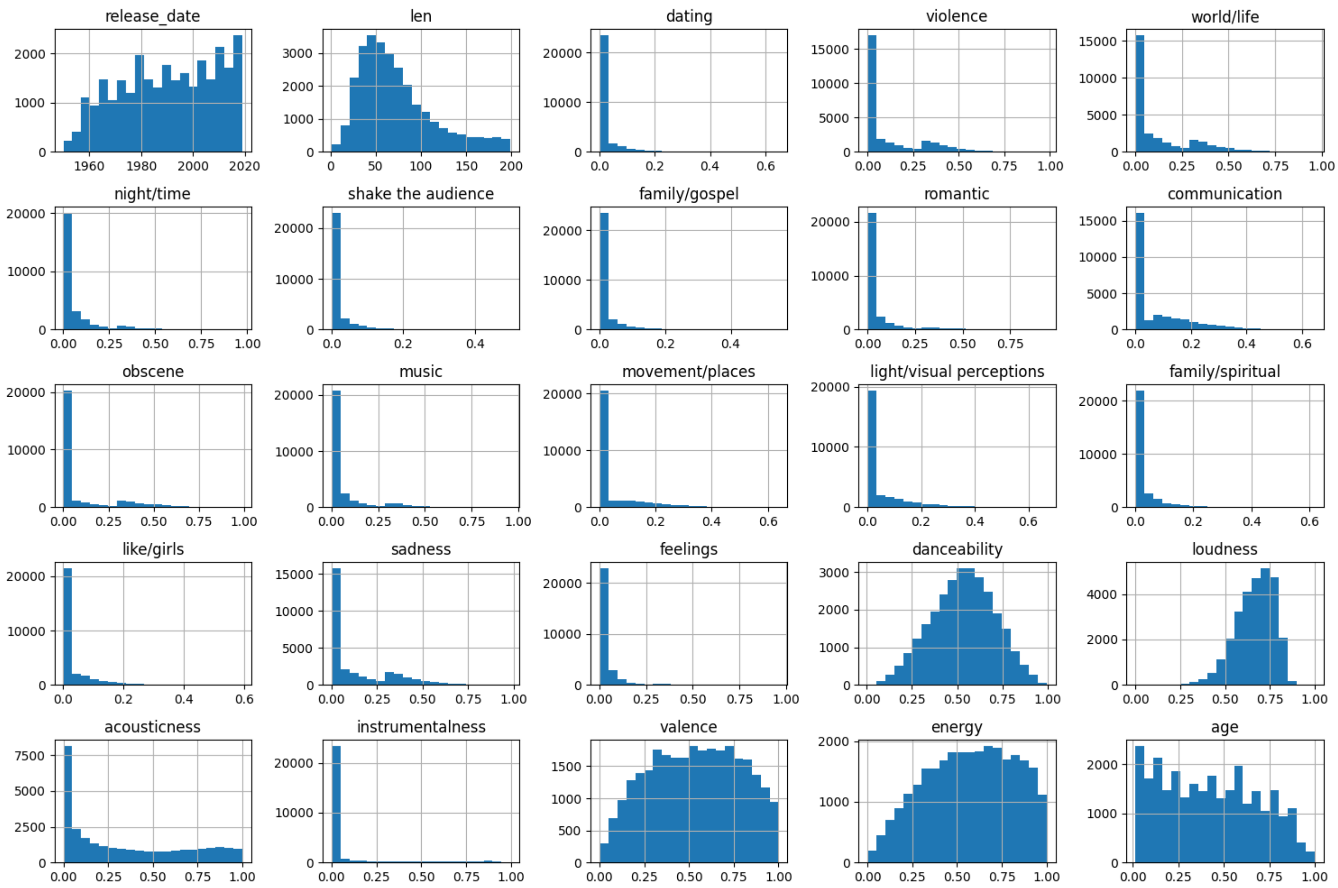
```
# For categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
categorical_cols = categorical_cols[categorical_cols != 'lyrics']
for col in categorical_cols:
    print(f'{col} unique values: {df[col].nunique()}')
    print(df[col].value_counts().head())  # Show top 5 most frequent catego

# Data appears to be mostly numerical types with the majority in float64 ty
# Genre, Lyrics, and Topic.
# For numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
print(df[numerical_cols].describe())
```

```
artist_name unique values: 5426
artist_name
johnny cash        190
ella fitzgerald    188
dean martin        146
willie nelson      131
george jones       107
Name: count, dtype: int64
track_name unique values: 23689
track_name
tonight        17
stay           15
hold on        15
without you    14
home           13
Name: count, dtype: int64
genre unique values: 7
genre
pop        7042
country    5445
blues      4604
rock       4034
jazz       3845
Name: count, dtype: int64
topic unique values: 8
topic
sadness      6096
violence     5710
world/life   5420
obscene      4882
music        2303
Name: count, dtype: int64
```

```
       release_date           len        dating      violence    world/life
count  28372.000000  28372.000000  28372.000000  28372.000000  28372.000000
mean    1990.236888     73.028444      0.021112      0.118396      0.120973
std       18.487463     41.829831      0.052370      0.178684      0.172200
min     1950.000000      1.000000      0.000291      0.000284      0.000291
25%     1975.000000     42.000000      0.000923      0.001120      0.001170
50%     1991.000000     63.000000      0.001462      0.002506      0.006579
75%     2007.000000     93.000000      0.004049      0.192608      0.197793
max     2019.000000    199.000000      0.647706      0.981781      0.962105

         night/time  shake the audience  family/gospel      romantic  \
count  28372.000000        28372.000000   28372.000000  28372.000000   
mean       0.057387            0.017422       0.017045      0.048681   
std        0.111923            0.040670       0.041966      0.106095   
min        0.000289            0.000284       0.000289      0.000284   
25%        0.001032            0.000993       0.000923      0.000975   
50%        0.001949            0.001595       0.001504      0.001754   
75%        0.065842            0.010002       0.004785      0.042301   
max        0.973684            0.497463       0.545303      0.940789   

       communication  ...    like/girls       sadness      feelings  \
count   28372.000000  ...  28372.000000  28372.000000  28372.000000   
mean        0.076680  ...      0.028057      0.129389      0.030996   
std         0.109538  ...      0.058473      0.181143      0.071652   
min         0.000291  ...      0.000284      0.000284      0.000289   
25%         0.001144  ...      0.000975      0.001144      0.000993   
50%         0.002632          0.001595      0.005263      0.001754
```

We will now look at histograms of our 25 numerical variables to get a sense of how they are distributed. These histograms display count values so we can see if certain patterns emerge and this will help us find correlations between variables.

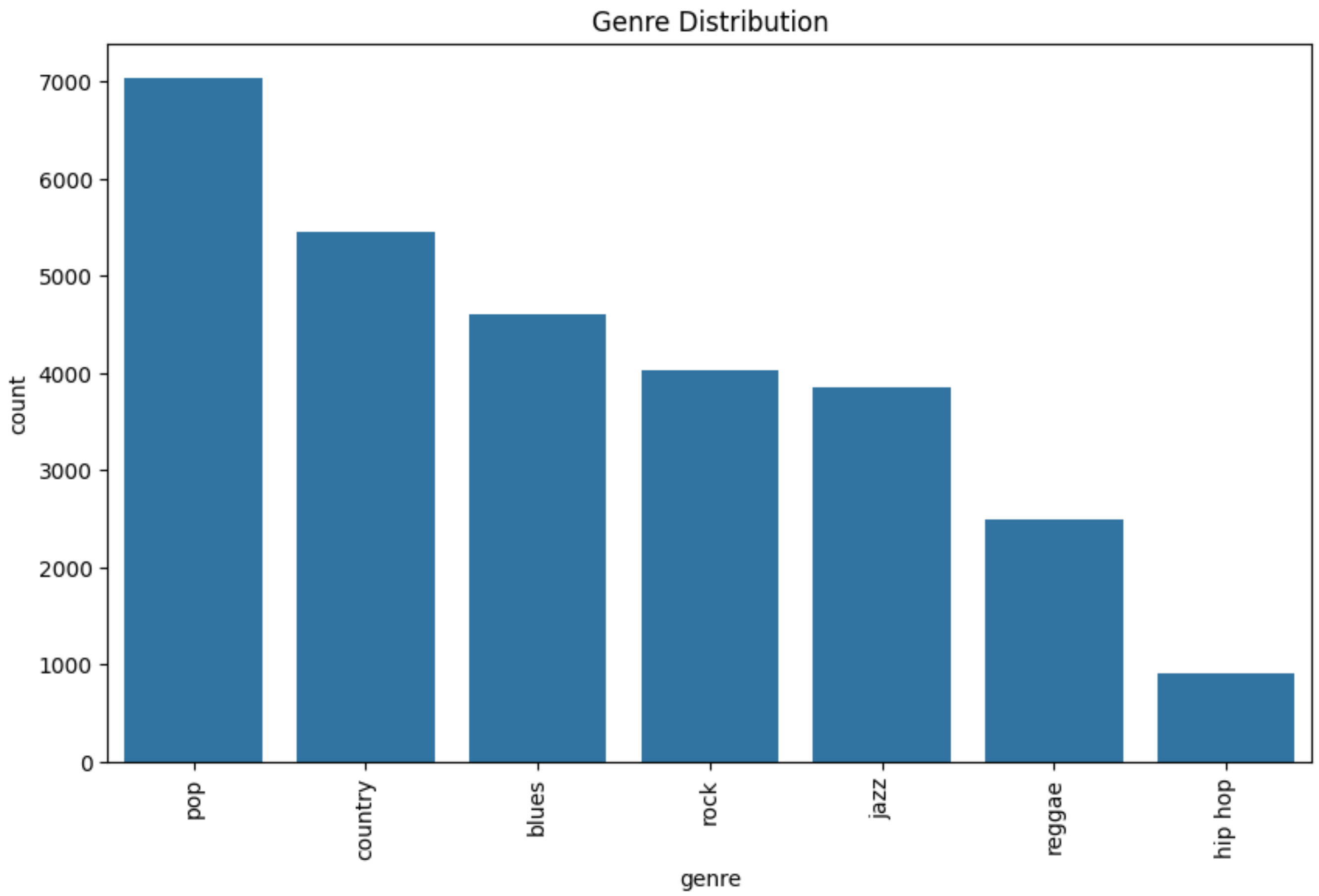
```
# Histogram for numerical columbns
df[numerical_cols].hist(bins=20, figsize=(15, 10))
plt.tight_layout()
plt.show()
```

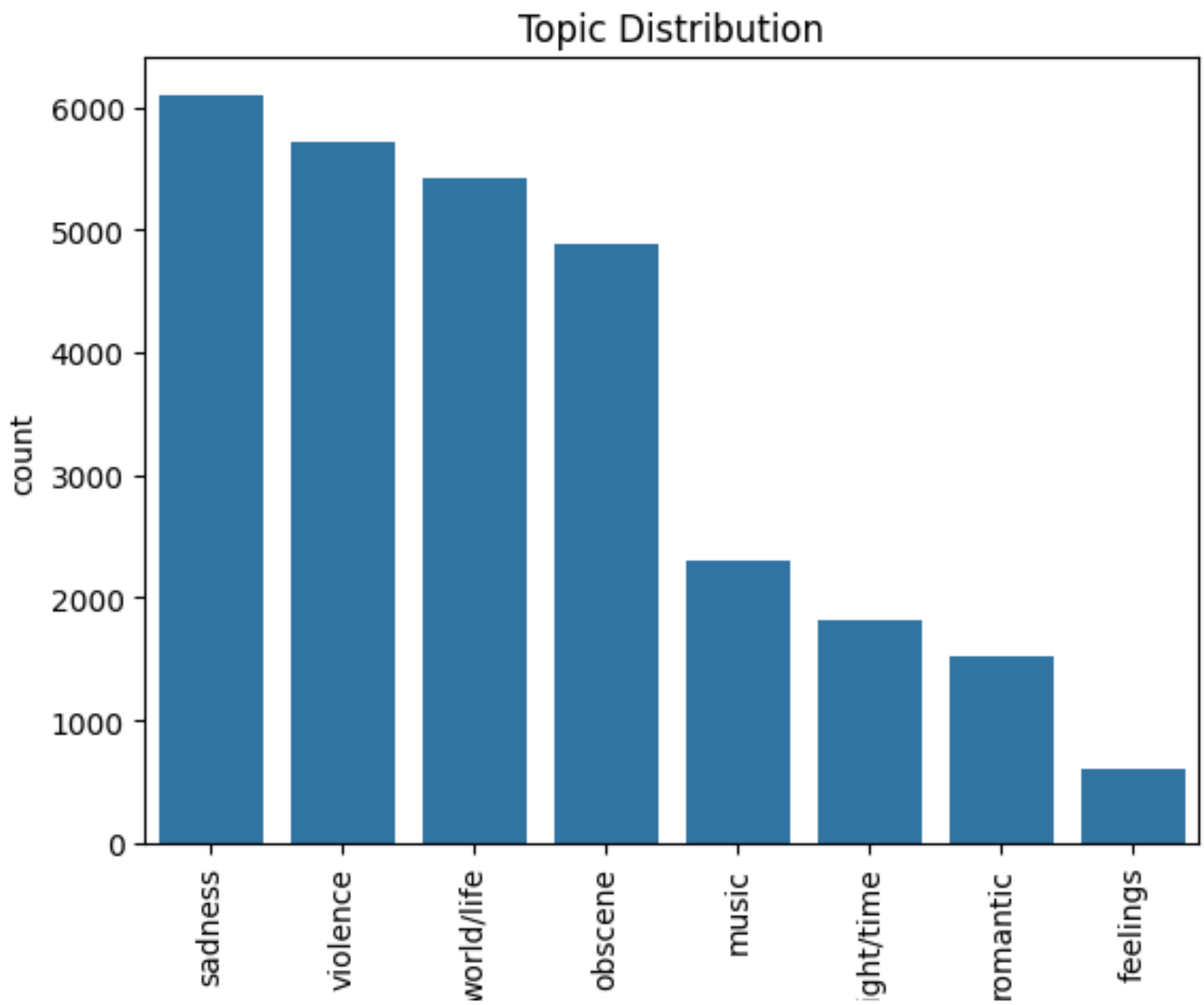


A few key patterns emerge from these histograms. Length is right tailed with a mean of about 50, and release date appears to have more songs from newer years. Danceability, valence, and energy have means of about 0.5 and have even tails on both sides. The bulk of the variables that contain emotional connotations of songs such as world/life, romantic, and sadness have very high means at just above 0, and have few values outside of this range.

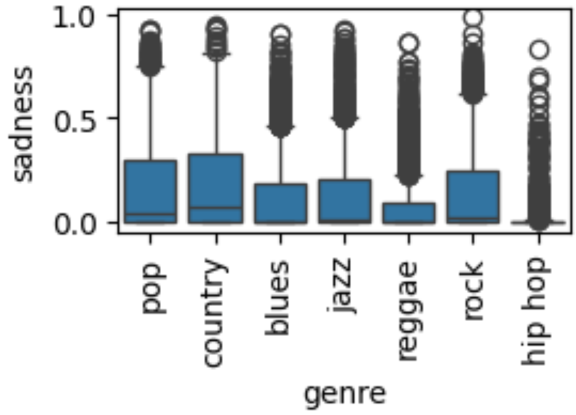
We will now look at barplots of the counts of each genre and song topic. Both genre and topic are expected to categorize similar songs together as differences between musical genres are quite large and users are predicted to generally like similar genre songs.

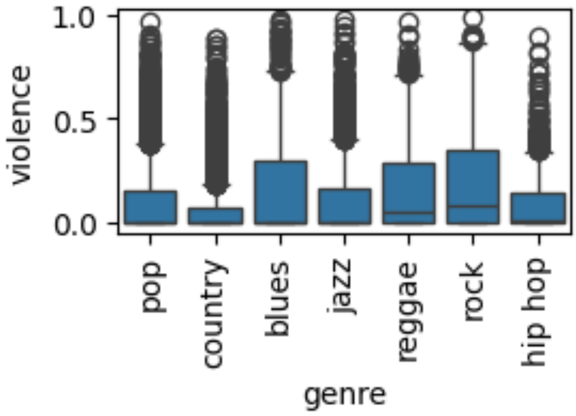
```
# Plot the distribution of genres in the dataset
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='genre', order=df['genre'].value_counts().index)
plt.title('Genre Distribution')  # Set plot title
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()

# Plot the distribution of topics in the dataset
sns.countplot(data=df, x='topic', order=df['topic'].value_counts().index)
plt.title('Topic Distribution')  # Set plot title
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()

# Create a boxplot to visualize the distribution of 'sadness' scores across
plt.figure(figsize=(6, 3))
plt.subplot(2, 2, 1)  # Specify subplot location (1st plot in a 2x2 layout)
sns.boxplot(data=df, x='genre', y='sadness')  # Boxplot for sadness by genr
plt.xticks(rotation=90)  # Rotate x-axis labels
plt.show()

# Create a boxplot to visualize the distribution of 'violence' scores acros
plt.figure(figsize=(6, 3))
plt.subplot(2, 2, 2)  # Specify subplot location (2nd plot in a 2x2 layout)
sns.boxplot(data=df, x='genre', y='violence')  # Boxplot for violence by ge
plt.xticks(rotation=90)  # Rotate x-axis labels
plt.show()

# Create a boxplot to visualize the distribution of 'world/life' scores acr
plt.figure(figsize=(6, 3))
plt.subplot(2, 2, 3)  # Specify subplot location (3rd plot in a 2x2 layout)
sns.boxplot(data=df, x='genre', y='world/life')  # Boxplot for world/life b
plt.xticks(rotation=90)  # Rotate x-axis labels
plt.show()

```

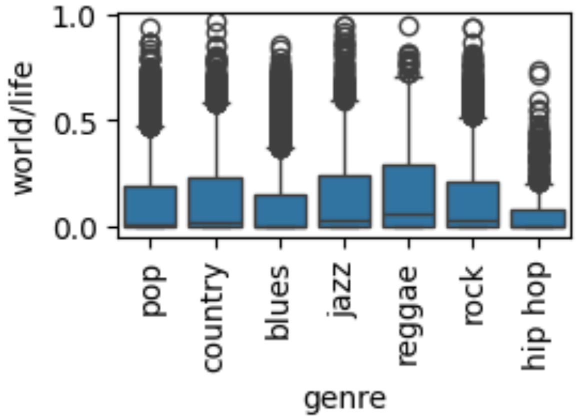

Genre Distribution
count
7000
6000
5000
4000
3000
2000
1000
0
pop
country
blues
rock
jazz
reggae
hip hop
genre


Topic Distribution
count
6000
5000
4000
3000
2000
1000
0
sadness
violence
world/life
obscene
music
ight/time
romantic
feelings

topic


1.0
0.5
0.0
sadness
pop
country
blues
jazz
reggae
rock
hip hop
genre

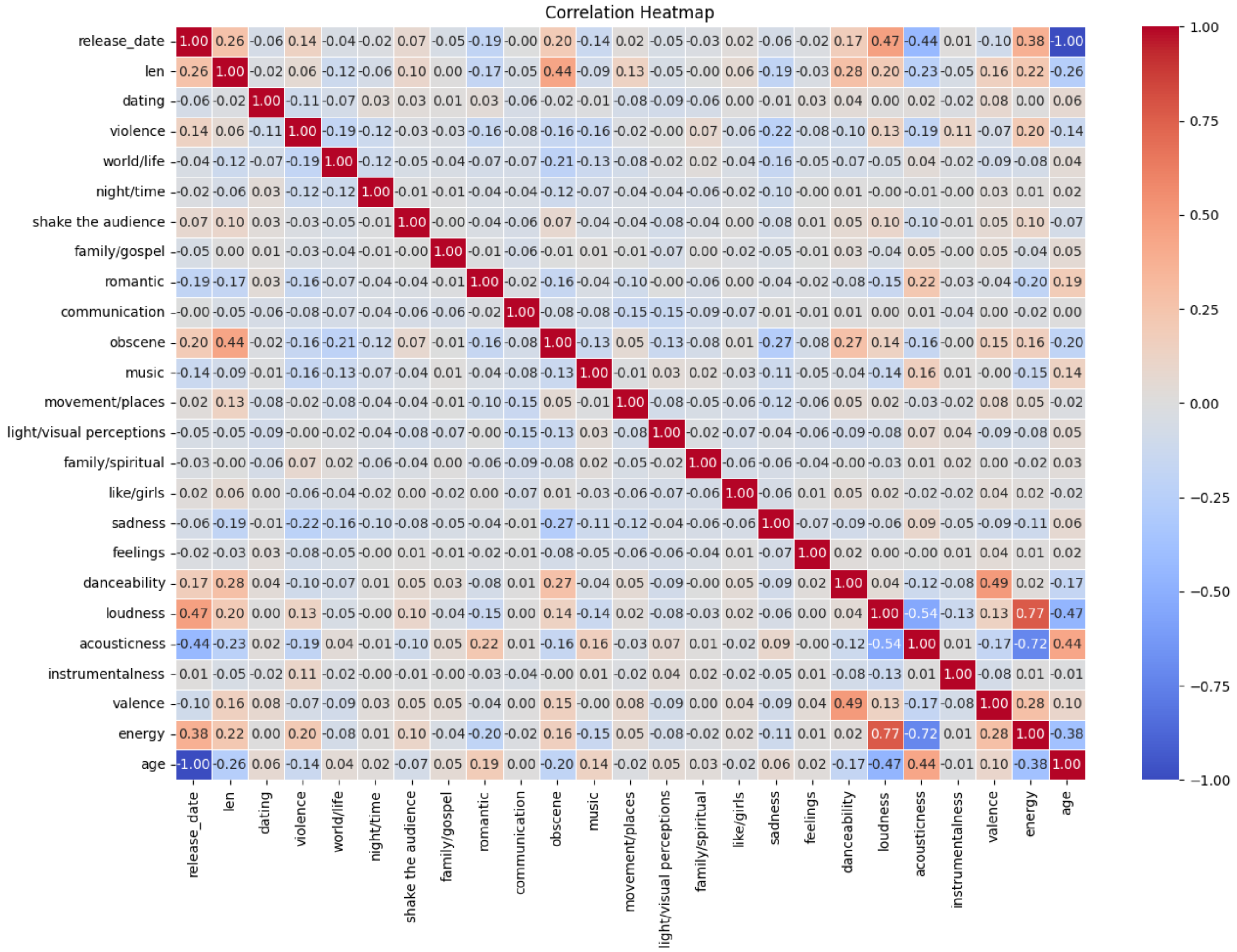

1.0
0.5
0.0
violence
pop
country
blues
jazz
reggae
rock
hip hop
genre


1.0
0.5
0.0
world/life
pop
country
blues
jazz
reggae
rock
hip hop
genre

As can be seen, pop is the top category with a difference of about 1500 from the second place country. Blues, Jazz, and Rock have similar counts and reggae and hip-hop have much less. In terms of topic there is a big difference in the values of the top 4 topics and the bottom 4. The top 4: sadness, violence, world/life, and obscene have values around 4800-6000 and the bottom 4 are all below 2500. We can generally expect to see these top genres and topics being recommended more by our NLP algorithm.

We will now look at a correlation matrix of all numeric values in our dataset. We will look for variables for strong positive and negative correlations as these relationships will be impactful in interpreting the results of our song interpreter.

```
# Correlation matrix
plt.figure(figsize=(15, 10))
corr = df[numerical_cols].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

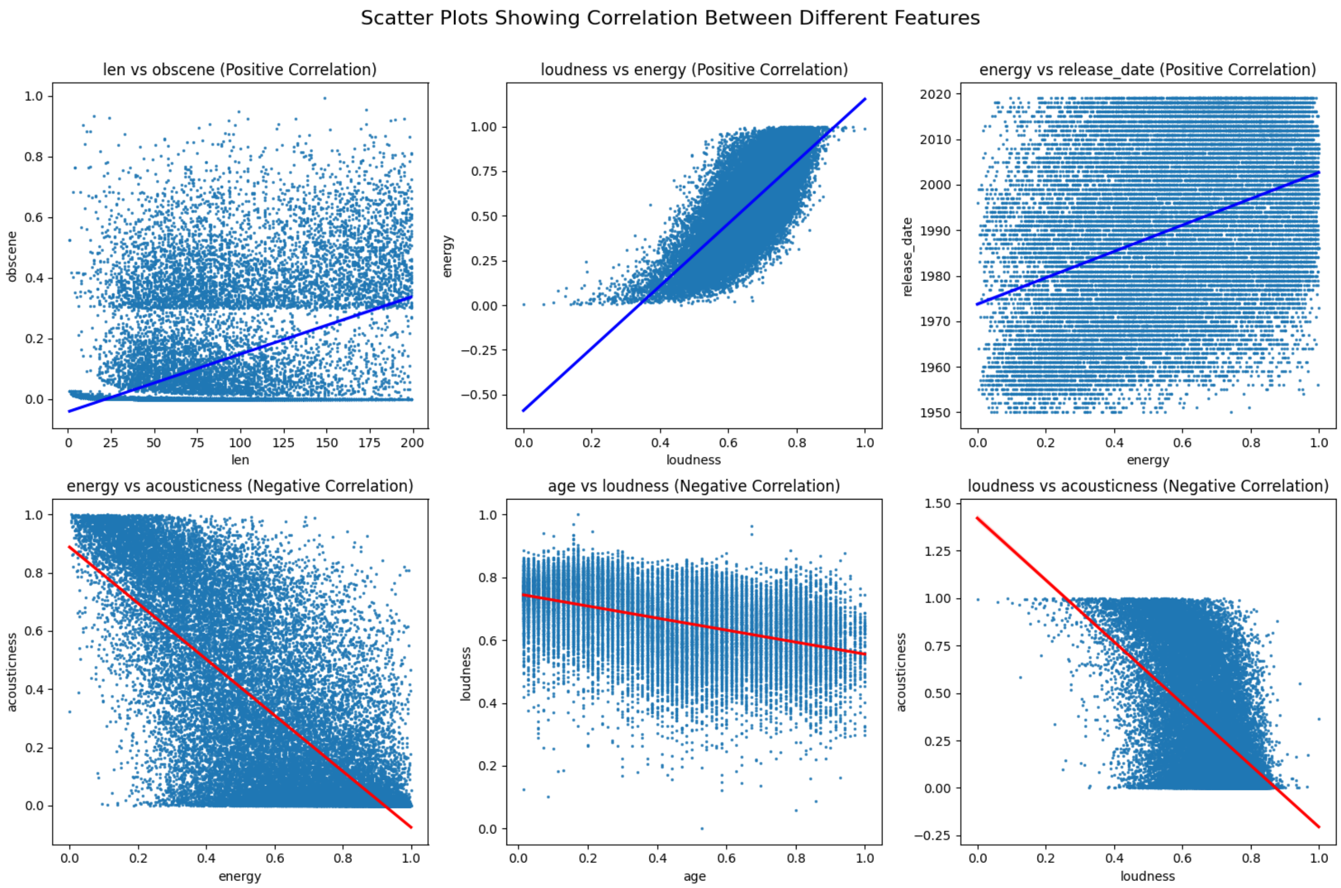

Correlation Heatmap
release_date 1.00 0.26 -0.06 0.14 -0.04 -0.02 0.07 -0.05 -0.19 -0.00 0.20 -0.14 0.02 -0.05 -0.03 0.02 -0.06 -0.02 0.17 0.47 -0.44 0.01 -0.10 0.38 -1.00
len 0.26 1.00 -0.02 0.06 -0.12 -0.06 0.10 0.00 -0.17 -0.05 0.44 -0.09 0.13 -0.05 -0.00 0.06 -0.19 -0.03 0.28 0.20 -0.23 -0.05 0.16 0.22 -0.26
dating -0.06 -0.02 1.00 -0.11 -0.07 0.03 0.03 0.01 0.03 -0.06 -0.02 -0.01 -0.08 -0.09 -0.06 0.00 -0.01 0.03 0.04 0.00 0.02 -0.02 0.08 0.00 0.06
violence 0.14 0.06 -0.11 1.00 -0.19 -0.12 -0.03 -0.03 -0.16 -0.08 -0.16 -0.16 -0.02 -0.00 0.07 -0.06 -0.22 -0.08 -0.10 0.13 -0.19 0.11 -0.07 0.20 -0.14
world/life -0.04 -0.12 -0.07 -0.19 1.00 -0.12 -0.05 -0.04 -0.07 -0.07 -0.21 -0.13 -0.08 -0.02 0.02 -0.04 -0.16 -0.05 -0.07 -0.05 0.04 -0.02 -0.09 -0.08 0.04
night/time -0.02 -0.06 0.03 -0.12 -0.12 1.00 -0.01 -0.01 -0.04 -0.04 -0.12 -0.07 -0.04 -0.04 -0.06 -0.02 -0.10 -0.00 0.01 -0.00 -0.01 -0.00 0.03 0.01 0.02
shake the audience 0.07 0.10 0.03 -0.03 -0.05 -0.01 1.00 -0.00 -0.04 -0.06 0.07 -0.04 -0.04 -0.08 -0.04 0.00 -0.08 0.01 0.05 0.10 -0.10 -0.01 0.05 0.10 -0.07
family/gospel -0.05 0.00 0.01 -0.03 -0.04 -0.01 -0.00 1.00 -0.01 -0.06 -0.01 0.01 -0.01 -0.07 0.00 -0.02 -0.05 -0.01 0.03 -0.04 0.05 -0.00 0.05 -0.04 0.05
romantic -0.19 -0.17 0.03 -0.16 -0.07 -0.04 -0.04 -0.01 1.00 -0.02 -0.16 -0.04 -0.10 -0.00 -0.06 0.00 -0.04 -0.02 -0.08 -0.15 0.22 -0.03 -0.04 -0.20 0.19
communication -0.00 -0.05 -0.06 -0.08 -0.07 -0.04 -0.06 -0.06 -0.02 1.00 -0.08 -0.08 -0.15 -0.15 -0.09 -0.07 -0.01 -0.01 0.01 0.00 0.01 -0.04 0.00 -0.02 0.00
obscene 0.20 0.44 -0.02 -0.16 -0.21 -0.12 0.07 -0.01 -0.16 -0.08 1.00 -0.13 0.05 -0.13 -0.08 0.01 -0.27 -0.08 0.27 0.14 -0.16 -0.00 0.15 0.16 -0.20
music -0.14 -0.09 -0.01 -0.16 -0.13 -0.07 -0.04 0.01 -0.04 -0.08 -0.13 1.00 -0.01 0.03 0.02 -0.03 -0.11 -0.05 -0.04 -0.14 0.16 0.01 -0.00 -0.15 0.14
movement/places 0.02 0.13 -0.08 -0.02 -0.08 -0.04 -0.04 -0.01 -0.10 -0.15 0.05 -0.01 1.00 -0.08 -0.05 -0.06 -0.12 -0.06 0.05 0.02 -0.03 -0.02 0.08 0.05 -0.02
light/visual perceptions -0.05 -0.05 -0.09 -0.00 -0.02 -0.04 -0.08 -0.07 -0.00 -0.15 -0.13 0.03 -0.08 1.00 -0.02 -0.07 -0.04 -0.06 -0.09 -0.08 0.07 0.04 -0.09 -0.08 0.05
family/spiritual -0.03 -0.00 -0.06 0.07 0.02 -0.06 -0.04 0.00 -0.06 -0.09 -0.08 0.02 -0.05 -0.02 1.00 -0.06 -0.06 -0.04 -0.00 -0.03 0.01 0.02 0.00 -0.02 0.03
like/girls 0.02 0.06 0.00 -0.06 -0.04 -0.02 0.00 -0.02 0.00 -0.07 0.01 -0.03 -0.06 -0.07 -0.06 1.00 -0.06 0.01 0.05 0.02 -0.02 -0.02 0.04 0.02 -0.02
sadness -0.06 -0.19 -0.01 -0.22 -0.16 -0.10 -0.08 -0.05 -0.04 -0.01 -0.27 -0.11 -0.12 -0.04 -0.06 -0.06 1.00 -0.07 -0.09 -0.06 0.09 -0.05 -0.09 -0.11 0.06
feelings -0.02 -0.03 0.03 -0.08 -0.05 -0.00 0.01 -0.01 -0.02 -0.01 -0.08 -0.05 -0.06 -0.06 -0.04 0.01 -0.07 1.00 0.02 0.00 -0.00 0.01 0.04 0.01 0.02
danceability 0.17 0.28 0.04 -0.10 -0.07 0.01 0.05 0.03 -0.08 0.01 0.27 -0.04 0.05 -0.09 -0.00 0.05 -0.09 0.02 1.00 0.04 -0.12 -0.08 0.49 0.02 -0.17
loudness 0.47 0.20 0.00 0.13 -0.05 -0.00 0.10 -0.04 -0.15 0.00 0.14 -0.14 0.02 -0.08 -0.03 0.02 -0.06 0.00 0.04 1.00 -0.54 -0.13 0.13 0.77 -0.47
acousticness -0.44 -0.23 0.02 -0.19 0.04 -0.01 -0.10 0.05 0.22 0.01 -0.16 0.16 -0.03 0.07 0.01 -0.02 0.09 -0.00 -0.12 -0.54 1.00 0.01 -0.17 -0.72 0.44
instrumentalness 0.01 -0.05 -0.02 0.11 -0.02 -0.00 -0.01 -0.00 -0.03 -0.04 -0.00 0.01 -0.02 0.04 0.02 -0.02 -0.05 0.01 -0.08 -0.13 0.01 1.00 -0.08 0.01 -0.01
valence -0.10 0.16 0.08 -0.07 -0.09 0.03 0.05 0.05 -0.04 0.00 0.15 -0.00 0.08 -0.09 0.00 0.04 -0.09 0.04 0.49 0.13 -0.17 -0.08 1.00 0.28 0.10
energy 0.38 0.22 0.00 0.20 -0.08 0.01 0.10 -0.04 -0.20 -0.02 0.16 -0.15 0.05 -0.08 -0.02 0.02 -0.11 0.01 0.02 0.77 -0.72 0.01 0.28 1.00 -0.38
age -1.00 -0.26 0.06 -0.14 0.04 0.02 -0.07 0.05 0.19 0.00 -0.20 0.14 -0.02 0.05 0.03 -0.02 0.06 0.02 -0.17 -0.47 0.44 -0.01 0.10 -0.38 1.00
release_date len dating violence world/life night/time shake the audience family/gospel romantic communication obscene music movement/places light/visual perceptions family/spiritual like/girls sadness feelings danceability loudness acousticness instrumentalness valence energy age
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

We can see that the largest Positive Correlations are between: Len and Obscene, Loudness and Energy, and Danceability and Valence We can see that the largest Negative Correlations are between: Energy and Acousticness, Age and Loudness, and Loudness and Acousticness

Intuitively many of these make sense, loud songs are more likely to have more energy and acousticness would lead to quiet songs which would have a negative correlation with loudness. An interesting unexpected correlation is that longer songs seem more likely to be more obscene and also that the valence (positivity or negativity) of a song is correlated with its danceability.

We will now look at scatterplots with a best fit linear regression line of the top 3 positive and negative relationships based on our correlation matrix. We want to visualize how strong these relationships are as these will provide good insight into what factors we can expect to be most influential in our algorithm.

```
# Define the pairs for plotting
positive_pairs = [('len', 'obscene'), ('loudness', 'energy'), ('energy', 'r
negative_pairs = [('energy', 'acousticness'), ('age', 'loudness'), ('loudne

# Create a figure and axis for each plot
fig, axs = plt.subplots(2, 3, figsize=(15, 10))  # 2 rows, 3 columns for 6
fig.suptitle('Scatter Plots Showing Correlation Between Different Features'

# Plot positive correlations with line of best fit
for i, (x_col, y_col) in enumerate(positive_pairs):
    sns.regplot(x=x_col, y=y_col, data=df, ax=axs[0, i], scatter_kws={'s':
    axs[0, i].set_title(f'{x_col} vs {y_col} (Positive Correlation)')
    axs[0, i].set_xlabel(x_col)
    axs[0, i].set_ylabel(y_col)

# Plot negative correlations with line of best fit
for i, (x_col, y_col) in enumerate(negative_pairs):
    sns.regplot(x=x_col, y=y_col, data=df, ax=axs[1, i], scatter_kws={'s':
    axs[1, i].set_title(f'{x_col} vs {y_col} (Negative Correlation)')
    axs[1, i].set_xlabel(x_col)
    axs[1, i].set_ylabel(y_col)

# Adjust the layout
plt.tight_layout()
plt.subplots_adjust(top=0.9)  # Adjust title spacing

# Show the plot
plt.show()
```


Scatter Plots Showing Correlation Between Different Features
len vs obscene (Positive Correlation)
obscene
len
loudness vs energy (Positive Correlation)
energy
loudness
energy vs release_date (Positive Correlation)
release_date
energy
energy vs acousticness (Negative Correlation)
acousticness
energy
age vs loudness (Negative Correlation)
loudness
age
loudness vs acousticness (Negative Correlation)
acousticness
loudness

We can immediately see that loudness and energy has a very clear relationship and the regression line indicates a strong positive correlation. The negative relationships have very strong trend lines and pattern of points appear to support their accuracy. Loudness vs. Acousticness has a very strong negative correlation which makes sense as loudness, energy, and acousticness all are very closely related. (Len and Obscene) and (Energy Release_Date) have strong positive relationships but the pattern isn't quite as apparent without the best fit line.

We will now plot and examine a word cloud before doing our NLP process so we can visualize the most popular words in each song's lyrics. We expect songs with similar vocabularies to be recommened together.

```
# Join all lyrics and create a word cloud
all_lyrics = ' '.join(df['lyrics'])
wordcloud = WordCloud(width=800, height=400, background_color='white').gene

# Display the wordcloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

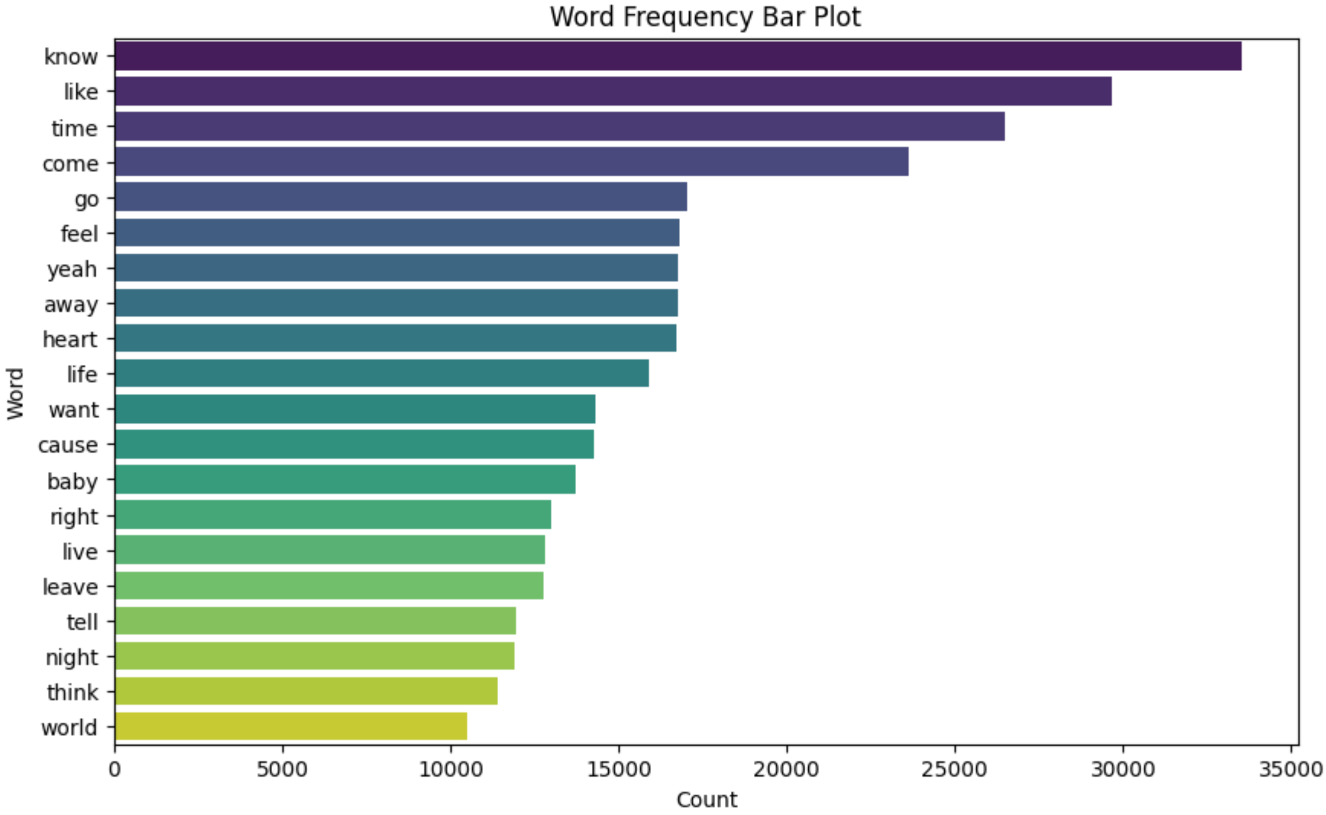
We can see based on the word cloud that Come, Know, Cause, Time and Want have the largest sizes respectively. The word cloud displays in proportion to their frequencies of occuring so we can visualize which words are the most common. This is important because a big portion of NLP analysis is looking at elements like the Cosine Similarity Score to see which songs have the most words in common.

We will now partition each word, remove stop words and plot a barplot of the most frequently occuring words.

```
from collections import Counter
stop_words = set(stopwords.words('english'))
```

```

# Tokenizing and removing stopwords
tokens = [word for word in all_lyrics.split() if word.lower() not in stop_w

# Get the most common words
common_words = Counter(tokens).most_common(20)
# Example common_words list

# Convert the list of tuples into a DataFrame
df_common_words = pd.DataFrame(common_words, columns=['Word', 'Count'])

# Sort by count in descending order for better visualization
df_common_words = df_common_words.sort_values(by='Count', ascending=False)

# Create the bar plot using seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x='Count', y='Word', data=df_common_words, palette='viridis')

# Set plot labels and title
plt.title('Word Frequency Bar Plot')
plt.xlabel('Count')
plt.ylabel('Word')

# Display the plot
plt.show()
```

```
<ipython-input-13-7721d693af20>:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.barplot(x='Count', y='Word', data=df_common_words, palette='viridis')
```

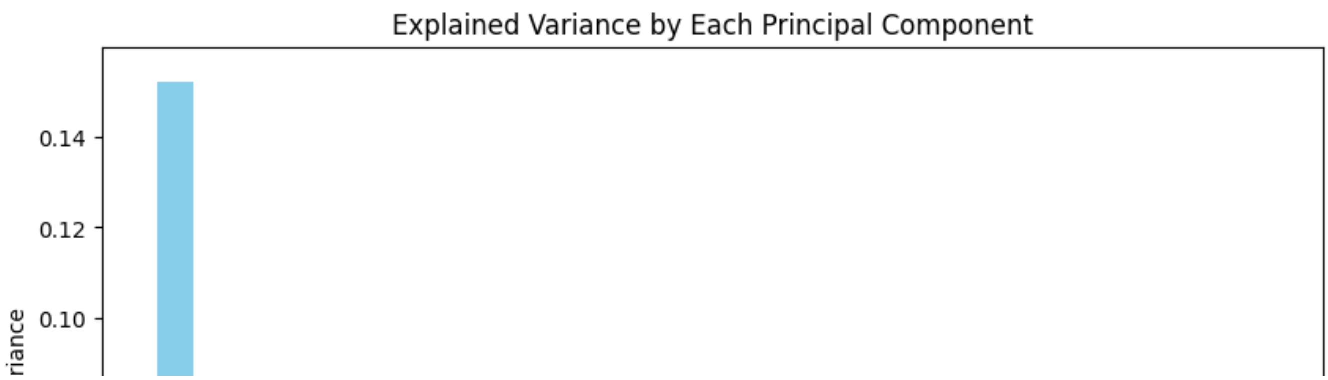


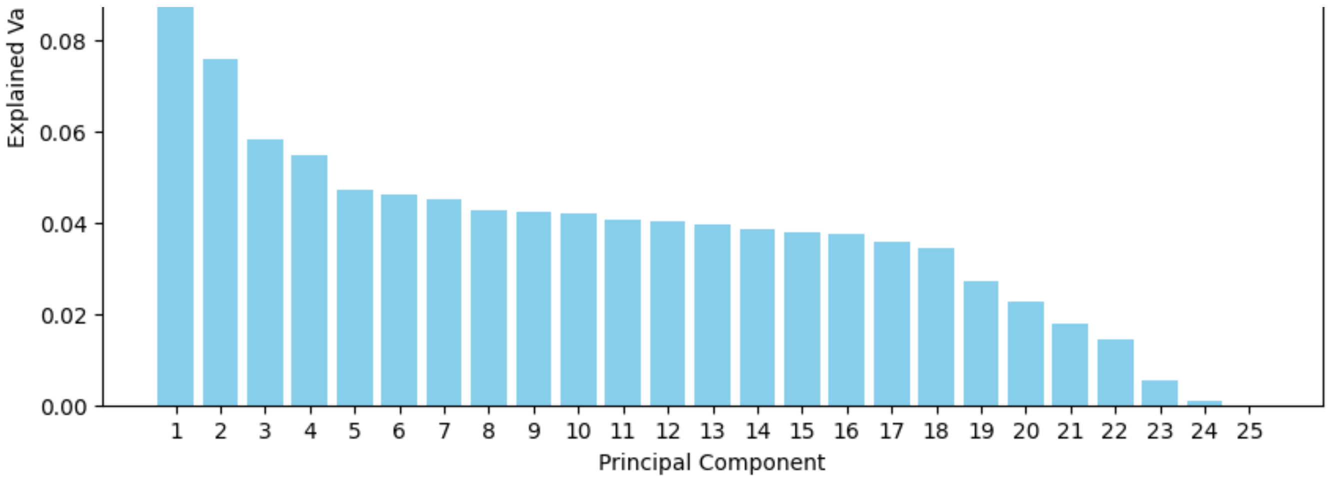
We can see that the words know, like, time, and come have by far the most occurences. We can see visualizing inspecting the words that these all seem like very common words musicians use in pop and country music in particular as well as music in general.

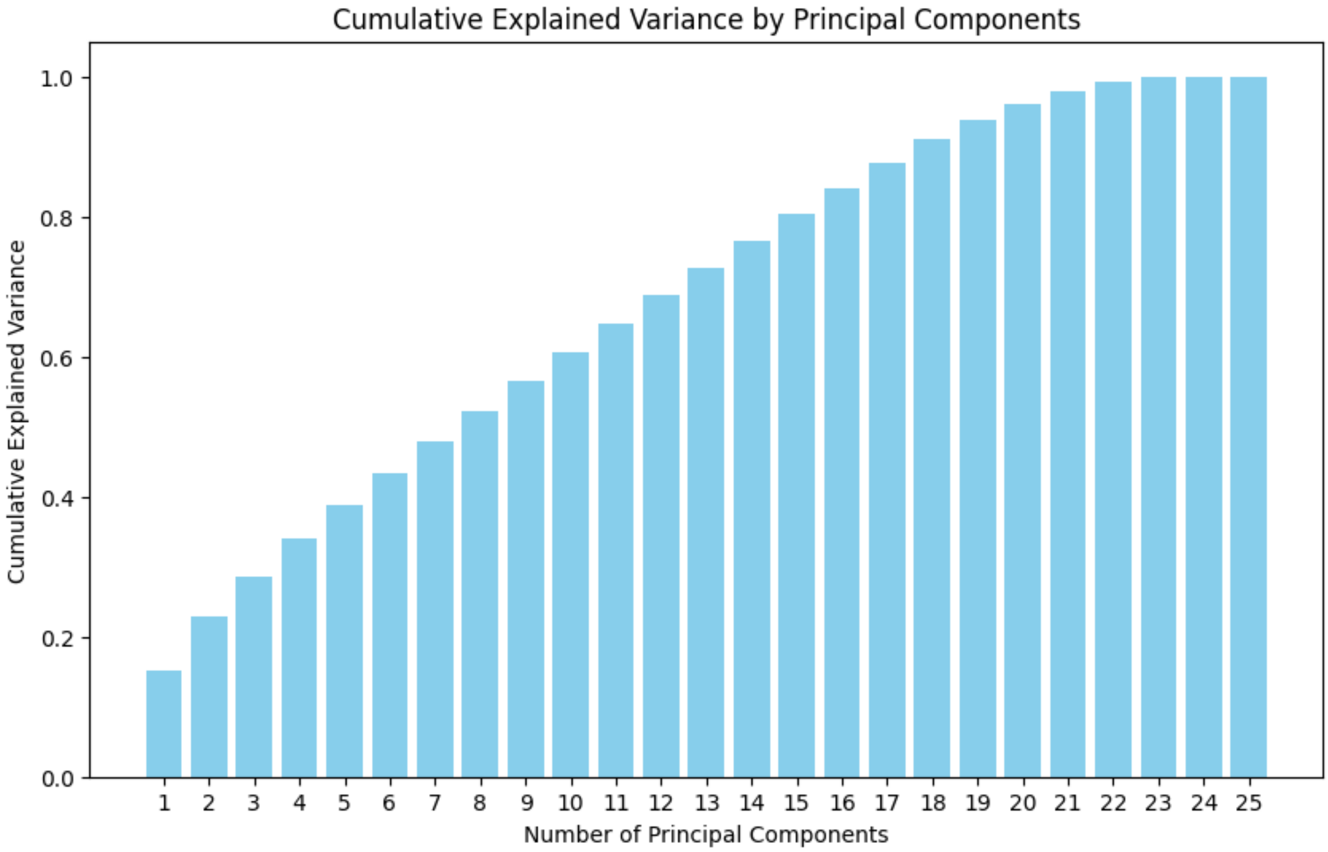
We will now perform Principle Component Analysis to examine which variables have the highest loadings on each principle component axis and how much each variable contributes to the cumulative variance.

```
import matplotlib.pyplot as plt
```

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Select numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
numerical_data = df[numerical_cols]

# Standardize the data
scaler = StandardScaler()
numerical_data_scaled = scaler.fit_transform(numerical_data)

# Apply PCA
pca = PCA()
pca.fit(numerical_data_scaled)

# Explained variance for each component
explained_variance = pca.explained_variance_ratio_

# Create a bar plot for explained variance
plt.figure(figsize=(10, 6))
plt.bar(range(1, len(explained_variance) + 1), explained_variance, color='s
plt.title('Explained Variance by Each Principal Component')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance')
plt.xticks(range(1, len(explained_variance) + 1))
plt.show()

# Cumulative explained variance (optional)
cumulative_explained_variance = np.cumsum(explained_variance)

# Create a bar plot for cumulative explained variance
plt.figure(figsize=(10, 6))
plt.bar(range(1, len(cumulative_explained_variance) + 1), cumulative_explai
plt.title('Cumulative Explained Variance by Principal Components')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.xticks(range(1, len(cumulative_explained_variance) + 1))
plt.show()
```




Explained Va
0.08
0.06
0.04
0.02
0.00
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Principal Component


Cumulative Explained Variance by Principal Components
Cumulative Explained Variance
1.0
0.8
0.6
0.4
0.2
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Number of Principal Components

We can see firstly based on the first plot that the first principle contributes by far the most to the dataset. PC1 has an explained variance of about 0.16 compared to just 0.08 for PC2. It also seems that PC3 - PC18 each contribute a similar amount of explained variance and linearly increase the total proportion explained. This indicates that to reach an explained cumulative variance of 0.9 we would still need about 18 PC components to do so.

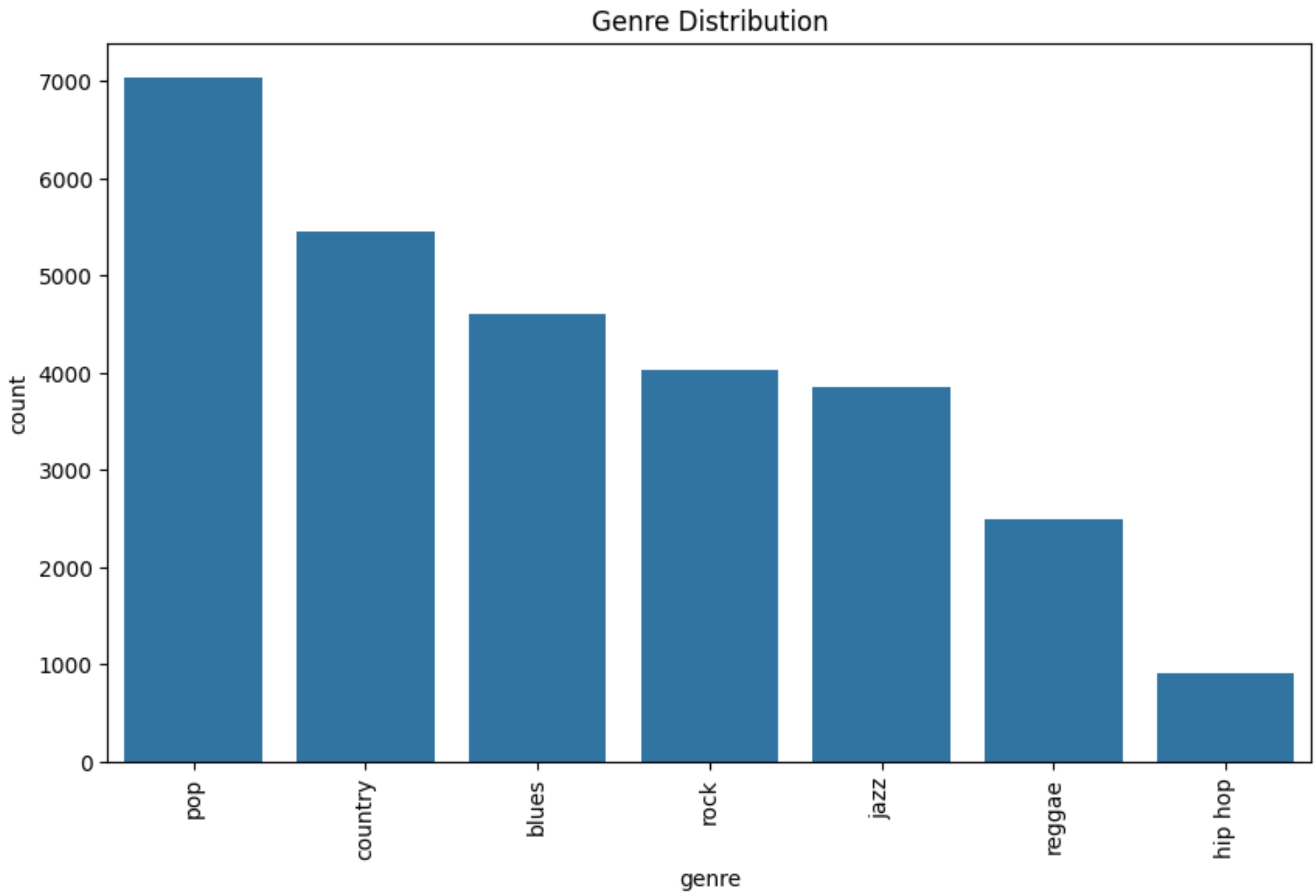
```
# Select numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
numerical_data = df[numerical_cols]

# Standardize the data
scaler = StandardScaler()
numerical_data_scaled = scaler.fit_transform(numerical_data)

# Apply PCA
pca = PCA()
pca.fit(numerical_data_scaled)

# Extract the loadings (components) for the first 3 principal components
components_df = pd.DataFrame(pca.components_, columns=numerical_cols)

# Print the first 3 components
print("First 3 Principal Components:")
for i in range(3):
    print(f"Principal Component {i+1}:")
    top_loadings = components_df.iloc[i].abs().nlargest(6).index
    for feature in top_loadings:
        print(f"{feature}: {components_df.iloc[i][feature]:.4f}")
    print("\n")
```

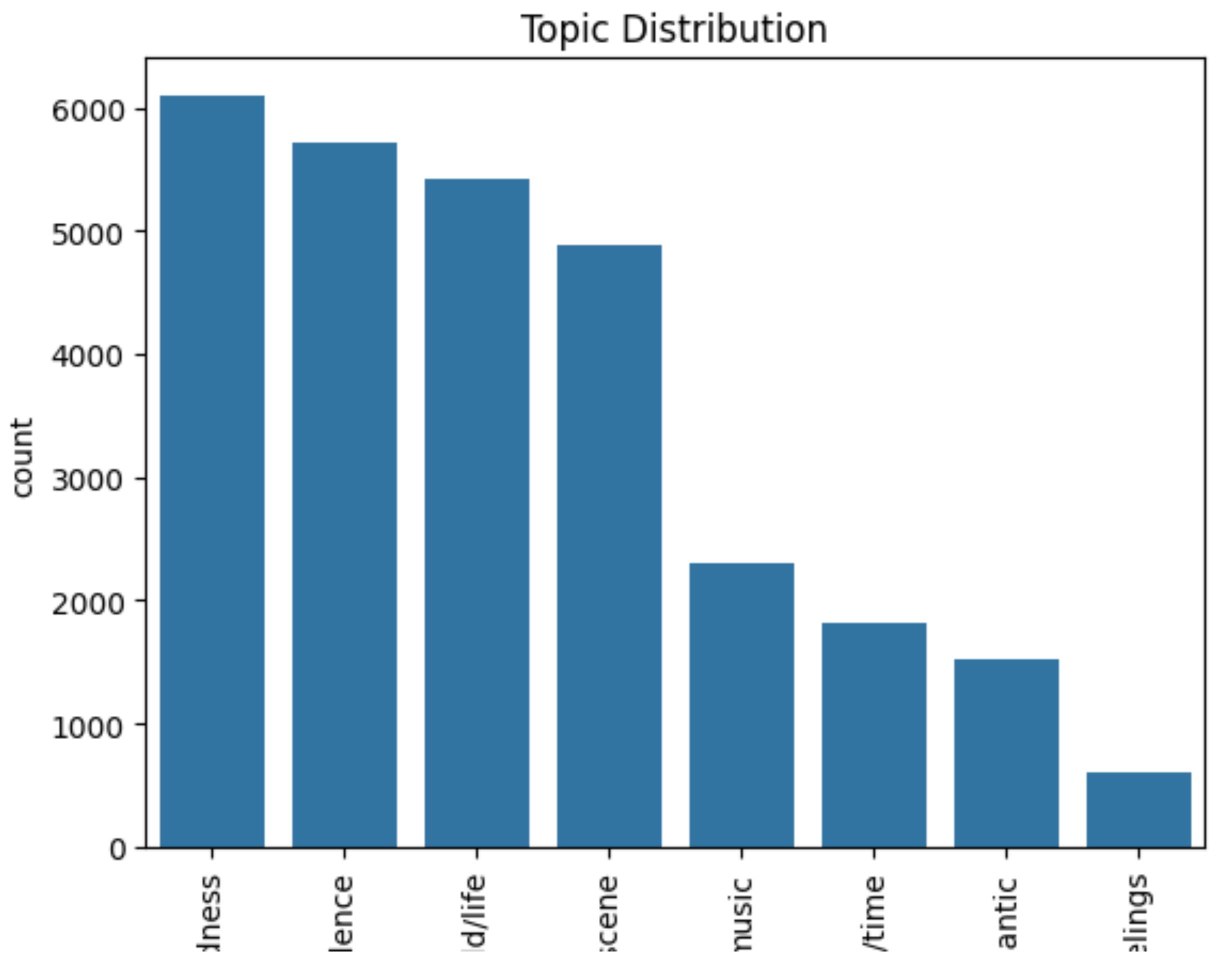
```
First 3 Principal Components:
Principal Component 1:
energy: 0.3964
release_date: 0.3946
age: -0.3946
acousticness: -0.3847
loudness: 0.3819
len: 0.2542


Principal Component 2:
valence: 0.4698
danceability: 0.4613
obscene: 0.3638
len: 0.2812
violence: -0.2595
age: 0.2137


Principal Component 3:
sadness: 0.3171
len: -0.2765
dating: 0.2704
communication: 0.2703
movement/places: -0.2572
obscene: -0.2534
```

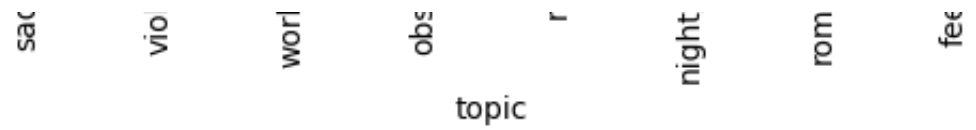
In the table above, we calculated the 5 highest absolute value loadings for the first 3 PC components and also displayed the variables alongside. We can see that for PC1 the first four variables energy, age, release_date, and acousticness have very similar loading scores and there is a drop off of about 0.13 to the 5th greatest loading score variable, len. Valenece and Danceability have the highest loadings for PC2 and Sadness has by far the greatest loading for PC3. These relationships between PC loadings are similar in many ways to some of the relationships in the correlation matrix above and indicate significant relationships between several predictor variables.

```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='genre', order=df['genre'].value_counts().index)
plt.title('Genre Distribution')
plt.xticks(rotation=90)
plt.show()
# Plot for other categorical columns if needed (e.g., artist_name, topic)
sns.countplot(data=df, x='topic', order=df['topic'].value_counts().index)
plt.title('Topic Distribution')
plt.xticks(rotation=90)
```

```
plt.show()
```


Genre Distribution
count
7000
6000
5000
4000
3000
2000
1000
0
pop
country
blues
rock
jazz
reggae
hip hop
genre


Topic Distribution
count
6000
5000
4000
3000
2000
1000
0



Recommender System

To give users generalized reconmendations of the most popular and most relevant songs, we will explore the world of recommender systems. We want to start off with a generalized recommender system that gives users the option to input a song in the Spotify music system and provides generalized reconmendations of the most similar songs to the given input. This recommender system follows the concept of content based methods in recommender systems, where we recommend new music songs to users based on the similarity of songs that the user has liked or previewed before.

```
# Step 1: We import necessary libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.neighbors import NearestNeighbors

# Step 2: We load the dataset
file_path = "/content/tcc_ceds_music.csv"  # Path to the uploaded file in C
df = pd.read_csv(file_path)

# Step 3: Preprocessing for our system

if 'Unnamed: 0' in df.columns:
    df.drop(columns=['Unnamed: 0'], inplace=True)

numeric_features = ['danceability', 'energy', 'acousticness', 'valence', 'l
X_numeric = df[numeric_features]

```

```
scaler = StandardScaler()
X_numeric_scaled = scaler.fit_transform(X_numeric)

df['processed_lyrics'] = df['lyrics'].apply(
    lambda x: ' '.join([word.lower() for word in str(x).split() if word.isa
)

vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
X_text = vectorizer.fit_transform(df['processed_lyrics'])

svd = TruncatedSVD(n_components=100, random_state=42)
X_text_reduced = svd.fit_transform(X_text)

# Step 4: We combine Numerical and Text Features
X_combined = np.hstack((X_numeric_scaled, X_text_reduced))

# Step 5: We build the Recommendation System
knn = NearestNeighbors(n_neighbors=11, metric='cosine')  # 11 because input
knn.fit(X_combined)


def recommend_songs(song_name, df, knn_model, combined_features):

    if song_name not in df['track_name'].values:
        print(f"Error: The song '{song_name}' does not exist in the dataset
        return None

    song_index = df[df['track_name'] == song_name].index[0]

    song_features = combined_features[song_index].reshape(1, -1)

    distances, indices = knn_model.kneighbors(song_features)

    similar_song_indices = indices.flatten()[1:]

    return df.iloc[similar_song_indices]

# Step 6: We test our Recommendation System
input_song = "i believe"
recommendations = recommend_songs(input_song, df, knn, X_combined)

if recommendations is not None:
    print(recommendations[['track_name', 'artist_name', 'danceability', 'er

```

```
                                              track_name       artist_name
23549                  i believe in the man in the sky     elvis presley
70                                          temptation      the platters
17243                                          pretend     nat king cole
17121                        love is the sweetest thing         ray noble
17223                                             snow  rosemary clooney
200                          try a little tenderness      the platters
7292                                        young love       sonny james
94                             september in the rain      the platters
17595  when you're smiling (the whole world smiles wi...     nat king cole
152                                 birth of the blues        perry como

       danceability    energy  acousticness   valence
23549      0.421640  0.293271      0.865462  0.415705
70         0.384815  0.322301      0.931727  0.389942
17243      0.333911  0.275253      0.930723  0.293075
17121      0.284090  0.309288      0.968875  0.322960
17223      0.338243  0.301279      0.873494  0.332234
200        0.367486  0.297275      0.774096  0.367271
7292       0.412975  0.350330      0.788152  0.325021
94         0.286256  0.235211      0.969879  0.241550
17595      0.392397  0.340320      0.860442  0.342539
152        0.375068  0.287265      0.894578  0.389942
```

In order to implement the recommender system, we first perform numerical feature scaling in order to obtain various TF-IDF values. Next, we reduce dimensionality by applying Singular Value Decomposition before finally building the recommender system. This ensures we have a working recommender system where we are able to input a song name and get the top ten most recommended most similar songs.

Recomender System, Using NLP

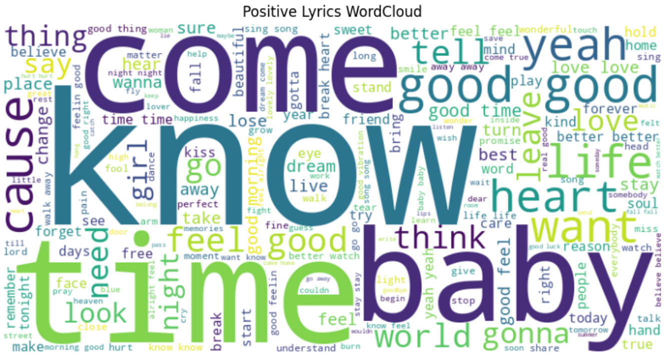
We have successfully built a recommender system using simpler content-based methods. Now, we want to see how incorporating NLP and sentiment analysis will enhance our recommendations.

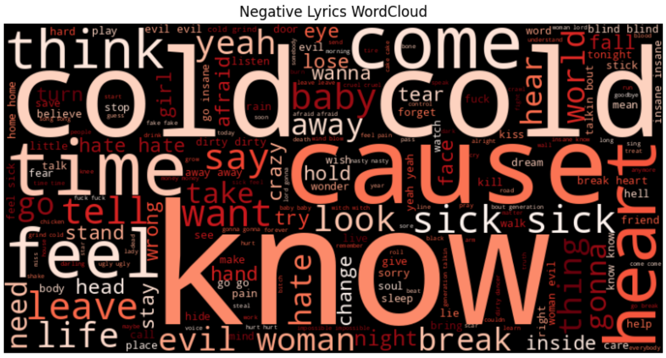
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors
```

```
from wordcloud import WordCloud
from textblob import TextBlob
import re
import string

# Step 1: Load the dataset
file_path = "/content/tcc_ceds_music.csv"  # Path to the uploaded file in
df = pd.read_csv(file_path)
df.drop(columns=['Unnamed: 0'], inplace=True)

# Create unique song identifier: artist - track
df['song_id'] = df['artist_name'] + " - " + df['track_name']

# Step 2: Clean the lyrics
def clean_lyrics(text):
    text = str(text).lower()
    text = re.sub(f"[{re.escape(string.punctuation)}]", "", text)
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'\s+', ' ', text)
    return text.strip()

df['clean_lyrics'] = df['lyrics'].apply(clean_lyrics)

# Step 3: Sentiment analysis
df['sentiment'] = df['clean_lyrics'].apply(lambda x: TextBlob(x).sentiment

# Show WordCloud for positive/negative lyrics
positive_lyrics = ' '.join(df[df['sentiment'] > 0.4]['clean_lyrics'])
negative_lyrics = ' '.join(df[df['sentiment'] < -0.4]['clean_lyrics'])

fig, ax = plt.subplots(1, 2, figsize=(16, 6))
wordcloud_pos = WordCloud(width=800, height=400, background_color='white')
wordcloud_neg = WordCloud(width=800, height=400, background_color='black',

ax[0].imshow(wordcloud_pos, interpolation='bilinear')
ax[0].axis('off')
ax[0].set_title("Positive Lyrics WordCloud")

ax[1].imshow(wordcloud_neg, interpolation='bilinear')
ax[1].axis('off')
ax[1].set_title("Negative Lyrics WordCloud")

plt.tight_layout()
plt.show()

# Step 4: TF-IDF + SVD
vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
X_text = vectorizer.fit_transform(df['clean_lyrics'])
```

```

svd = TruncatedSVD(n_components=100, random_state=42)
X_text_reduced = svd.fit_transform(X_text)

# Step 5: Scale numeric features (including sentiment)
numeric_features = ['danceability', 'energy', 'acousticness', 'valence',
                    'loudness', 'instrumentalness', 'age', 'sentiment']
X_numeric = df[numeric_features]

scaler = StandardScaler()
X_numeric_scaled = scaler.fit_transform(X_numeric)

# Step 6: Combine all features
X_combined = np.hstack((X_numeric_scaled, X_text_reduced))

# Step 7: Build the KNN model
knn = NearestNeighbors(n_neighbors=11, metric='cosine')
knn.fit(X_combined)

# Step 8: Song recommendation function using 'song_id'
def recommend_songs(song_id, df, knn_model, feature_matrix):
    if song_id not in df['song_id'].values:
        print(f"'{song_id}' not found in dataset.")
        return None

    song_index = df[df['song_id'] == song_id].index[0]
    song_vector = feature_matrix[song_index].reshape(1, -1)

    distances, indices = knn_model.kneighbors(song_vector)
    similar_indices = indices.flatten()[1:]  # exclude the input song itse

    recommendations = df.iloc[similar_indices].copy()
    return recommendations

# Step 9: Test the recommender
input_song_id = "nujabes - luv (sic.) pt3 (feat. shing02)"  # Replace with
recommendations = recommend_songs(input_song_id, df, knn, X_combined)

if recommendations is not None:
    print("Top 10 similar songs to:", input_song_id)
    display(recommendations[['song_id', 'danceability', 'energy', 'valence

    # Step 10: Visualize sentiment of recommended songs
    plt.figure(figsize=(10, 5))
    sns.barplot(x='song_id', y='sentiment', data=recommendations)
    plt.xticks(rotation=45, ha='right')
    plt.title('Sentiment Scores of Recommended Songs')
    plt.ylabel('Polarity (-1 to 1)')
```

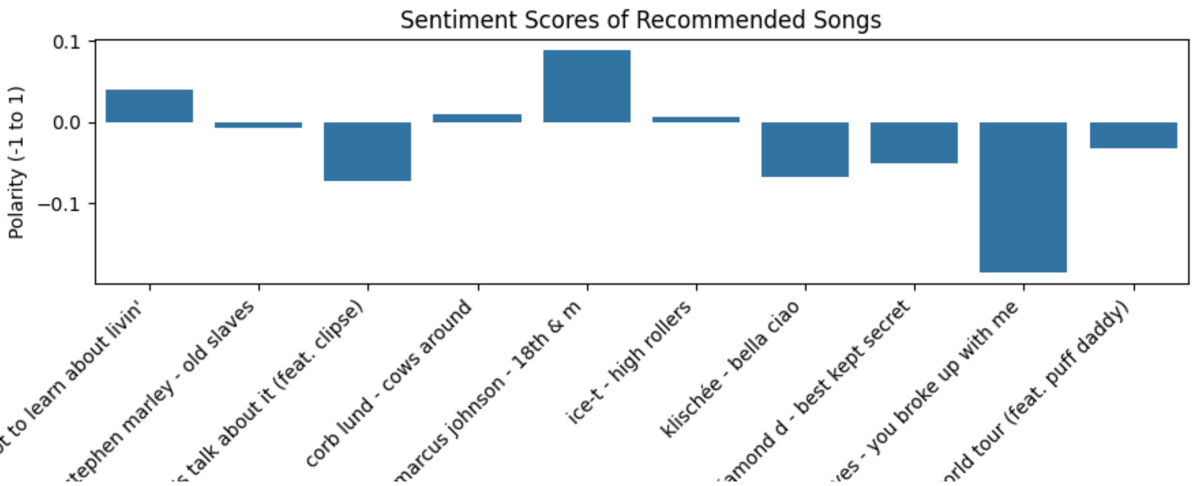
```
    plt.xlabel('Song (Artist - Track)')
    plt.tight_layout()
    plt.show()
```





Top 10 similar songs to: nujabes - luv (sic.) pt3 (feat. shing02)

	song_id	danceability	energy	valence	sentiment
11825	easton corbin - a lot to learn about livin'	0.818044	0.634623	0.927865	0.040246
22826	stephen marley - old slaves	0.877613	0.538524	0.939200	-0.007143
28252	jermaine dupri - let's talk about it (feat. cl...	0.939348	0.661651	0.815540	-0.072727
11938	corb lund - cows around	0.793133	0.624613	0.733100	0.009703
19832	marcus johnson - 18th & m	0.787718	0.676667	0.871187	0.088159
28017	ice-t - high rollers	0.879779	0.552539	0.875309	0.006571
20836	klischée - bella ciao	0.888444	0.810805	0.824815	-0.067857
27749	diamond d - best kept secret	0.868948	0.667657	0.912407	-0.050000
12328	walker hayes - you broke up with me	0.850536	0.699690	0.876340	-0.185746
28007	black rob - pd world tour (feat. puff daddy)	0.727066	0.603591	0.889736	-0.032222



Each song lyric is first assigned a sentiment polarity score ranging from -1 (very negative) to +1 (very positive). To capture the most meaningful language features, the top 5,000 important words from each lyric are extracted using TF-IDF. These high-dimensional features are then reduced to 100 components using Truncated SVD, allowing for efficient processing. In addition to lyrical data, numeric audio features such as danceability, energy, and valence are scaled and combined with the lyric embeddings to form a unified feature vector, referred to as $X_{combined}$. Using this combined representation, a K-Nearest Neighbors (KNN) model with cosine similarity is applied to find the 10 most similar songs based on both textual and acoustic characteristics, including track name, artist, danceability, energy, valence, and sentiment. Word clouds are generated to visualize word frequency patterns—one showing overall common words across all lyrics, and two others contrasting the vocabulary of positively vs. negatively scored lyrics. Finally, the system outputs the top 10 most similar tracks along with key metadata, and presents a bar chart of their sentiment scores to help assess the emotional similarity between the input song and the recommendations.

Methods/Results

The first step for our project was exploring and understanding the data set. We first got rid of any unnecessary columns, along with any NA data points of which there weren't many. After this, we created some histogram visuals to examine the distributions of our variables and look for any possible outliers that may need to be dealt with. We also created a correlation matrix to help us understand the hidden correlations within the data. This was incredibly helpful, as it allowed us to see that our data was not just randomly generated but ather it contains meaningful patterns that mirror what we'd expect based on our previous knowledge. For example, we were able to find and plot the negative relationship between acousticness and energy, which is exactly what we'd expect based on the contents of these variables. We then prepared our NLP by looking at a wordcloud of our data's 'lyrics' column and then creating a barplot of the most common words, after tokenizing and removing stop words. The last step before building our recommenders was performing a PCA analysis on the data to understand the influence of specific variables and further strengthen our argument of stastically significant relationships being present in the data. We found that age, energy, release_date, and acousticness contributed most to our PC1, putting them among our most influential variables and once again suggesting a strong presence of correlation within the data.

Our project used two recommendation systems. The first recommendation system that we implemented was the simpler content based recommendation system. For this recommendation system, we apply the concept of content-based filtering. Content-based filtering refers to recommendation systems that are built based on the content being investigated. This is achieved by calculating the similarity between various items and comparable items that the user has liked before. For example, if the user had previously shown a preference for item A, and item A and item B are similar, then based on content-based filtering, the recommendation system recommends item B to the user. In order to implement the content-based filtering recommender system for our Spotify Music Recommender project, we provide recommendations for a specific song based on features including danceability, energy, acousticness, and valence. To start, we first perform numerical feature scaling in order to obtain various TF-IDF values. Next, we reduce dimensionality by applying Singular Value Decomposition before we combine the numerical and text features together. Finally, we build the function for our recommender system, with the function taking in inputs including “song_name”, “df”, “knn_model”, and “combined_features”. In order to test the validity of our recommender system, we input a song from the Spotify music list called “i believe”, and we obtain the top ten most similar songs to “i believe” with values for each specifc song based on the features of danceability, energy, acousticness, and valence. We now have a working recommender system where we are able to input a song name and get the top ten most recommended most similar songs.

Conclusion

In this project, we successfully designed and implemented two **content-based music recommender systems** using a comprehensive dataset of over 28,000 Spotify tracks. Our objective was to generate meaningful song recommendations by analyzing both **audio-based features** and **lyrical content**.

The first system utilized standardized numeric features such as `danceability`, `energy`, `valence`, and others, combined with reduced TF-IDF vectors from lyrics. By applying **cosine similarity** through a K-Nearest Neighbors model, we generated recommendations based on musical and lyrical proximity. The second system expanded this foundation by incorporating **sentiment scores** derived from lyrics using **TextBlob**, enabling the model to account for emotional tone in the recommendation process.

Key takeaways include:

- **EDA (Exploratory Data Analysis)** revealed important trends: *Pop* was the dominant genre, and topics like *sadness* and *violence* were most prevalent. Strong correlations such as `energy ~ loudness` and `acousticness ~ loudness` aligned with musical intuition.
- **PCA** showed that dimensionality reduction was both necessary and effective. While the first principal component explained around 16% of the variance, nearly 18 components were needed to reach 90% cumulative variance.
- The **word cloud and frequency plots** confirmed that common emotional and expressive words dominate lyrical themes, with “know”, “like”, and “time” being the most frequent.
- Our recommendation system demonstrated strong practical value, effectively recommending emotionally and sonically similar tracks — as shown in the example for *“I Believe”* by Frankie Laine, which returned songs with similar sentiment and style from artists like *Nat King Cole* and *Elvis Presley*.

This project illustrates the power of combining **structured numerical data** with **unstructured text-based NLP** to build intelligent recommender systems. Moving forward, incorporating collaborative filtering or hybrid models could further improve personalization. Additionally, real-time user feedback could be integrated to dynamically tune recommendations.