Music Recommendation System with Natural Language Processing

Import Necessary Libraries

```
# 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
```

Load and Explore Dataset

```
file_path = "/content/tcc_ceds_music.csv"  # Path to the uploaded file in Cc
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
2	6	johnnie ray	cry	1950	pop	sweetheart 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
dtypes: float64(23), int64(3), object(5)
memory usage: 6.7+ MB
None
```

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

```
# Display first few rows and dataset info after Droping unnecessary colums
display(df.head())
display(df.info())
```

	artist_name	track_name	release_date	genre	lyrics	len	dating	viol
0	mukesh	mohabbat bhi jhoothi	1950	pop	hold time feel break feel untrue convince spea...	95	0.000598	0.06
1	frankie laine	i believe	1950	pop	believe drop rain fall grow believe darkest ni...	51	0.035537	0.09
2	johnnie ray	cry	1950	pop	sweetheart send letter goodbye secret feel bet...	24	0.002770	0.00
					kiss lips			

3	pérez prado	patricia	1950	pop	kiss lips want stroll charm mambo chacha merin...	54	0.048249	0.00
4	giorgos papadopoulos	apopse eida oneiro	1950	pop	till darling till matter know till dream live ...	48	0.001350	0.00

5 rows × 30 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28372 entries, 0 to 28371
Data columns (total 30 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   artist_name               28372 non-null  object 
 1   track_name                28372 non-null  object 
 2   release_date              28372 non-null  int64  
 3   genre                     28372 non-null  object 
 4   lyrics                    28372 non-null  object 
 5   len                       28372 non-null  int64  
 6   dating                    28372 non-null  float64
 7   violence                  28372 non-null  float64
 8   world/life                28372 non-null  float64
 9   night/time                28372 non-null  float64
 10  shake the audience        28372 non-null  float64
 11  family/gospel             28372 non-null  float64
 12  romantic                  28372 non-null  float64
 13  communication             28372 non-null  float64
 14  obscene                   28372 non-null  float64
 15  music                     28372 non-null  float64
 16  movement/places           28372 non-null  float64
 17  light/visual perceptions  28372 non-null  float64
 18  family/spiritual          28372 non-null  float64
 19  like/girls                28372 non-null  float64
 20  sadness                   28372 non-null  float64
 21  feelings                  28372 non-null  float64
 22  danceability              28372 non-null  float64
 23  loudness                  28372 non-null  float64
 24  acousticness              28372 non-null  float64
 25  instrumentalness          28372 non-null  float64
 26  valence                   28372 non-null  float64
 27  energy                    28372 non-null  float64
 28  topic                     28372 non-null  object 
 29  age                       28372 non-null  float64
dtypes: float64(23), int64(2), object(5)
memory usage: 6.5+ MB
None
```

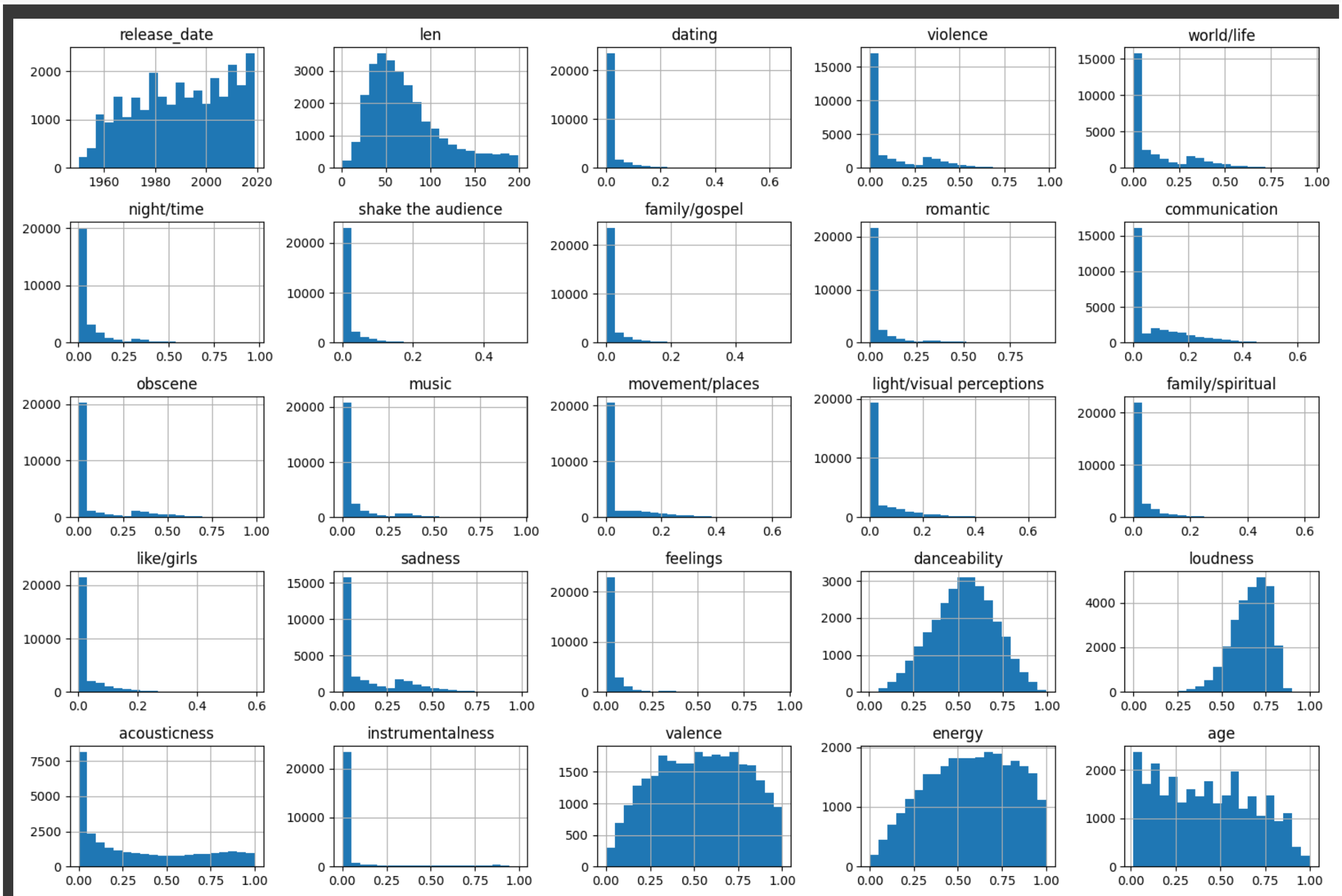
```
# Data appears to be mostly numerical types with the majority in float64 typ
# Genre, Lyrics, and Topic.
# For numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
print(df[numerical_cols].describe())
```

```
       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
75%         0.132136  ...      0.026622      0.235113      0.032622
max         0.645829  ...      0.594459      0.981424      0.958810

       danceability      loudness  acousticness  instrumentalness  \
count  28372.000000  28372.000000  2.837200e+04      28372.000000
mean       0.533348      0.665249  3.392347e-01          0.080049
std        0.173218      0.108434  3.267143e-01          0.211245
min        0.005415      0.000000  2.811248e-07          0.000000
25%        0.412975      0.595364  3.423598e-02          0.000000
50%        0.538612      0.679050  2.259028e-01          0.000085
75%        0.656666      0.749026  6.325298e-01          0.009335
max        0.993502      1.000000  1.000000e+00          0.996964

            valence        energy           age
count  28372.000000  28372.000000  28372.000000
mean       0.532864      0.569875      0.425187
std        0.250972      0.244385      0.264107
min        0.000000      0.000000      0.014286
25%        0.329143      0.380361      0.185714
50%        0.539365      0.580567      0.414286
75%        0.738252      0.772766      0.642857
max        1.000000      1.000000      1.000000

[8 rows x 25 columns]
```

```python
# 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 categories
```

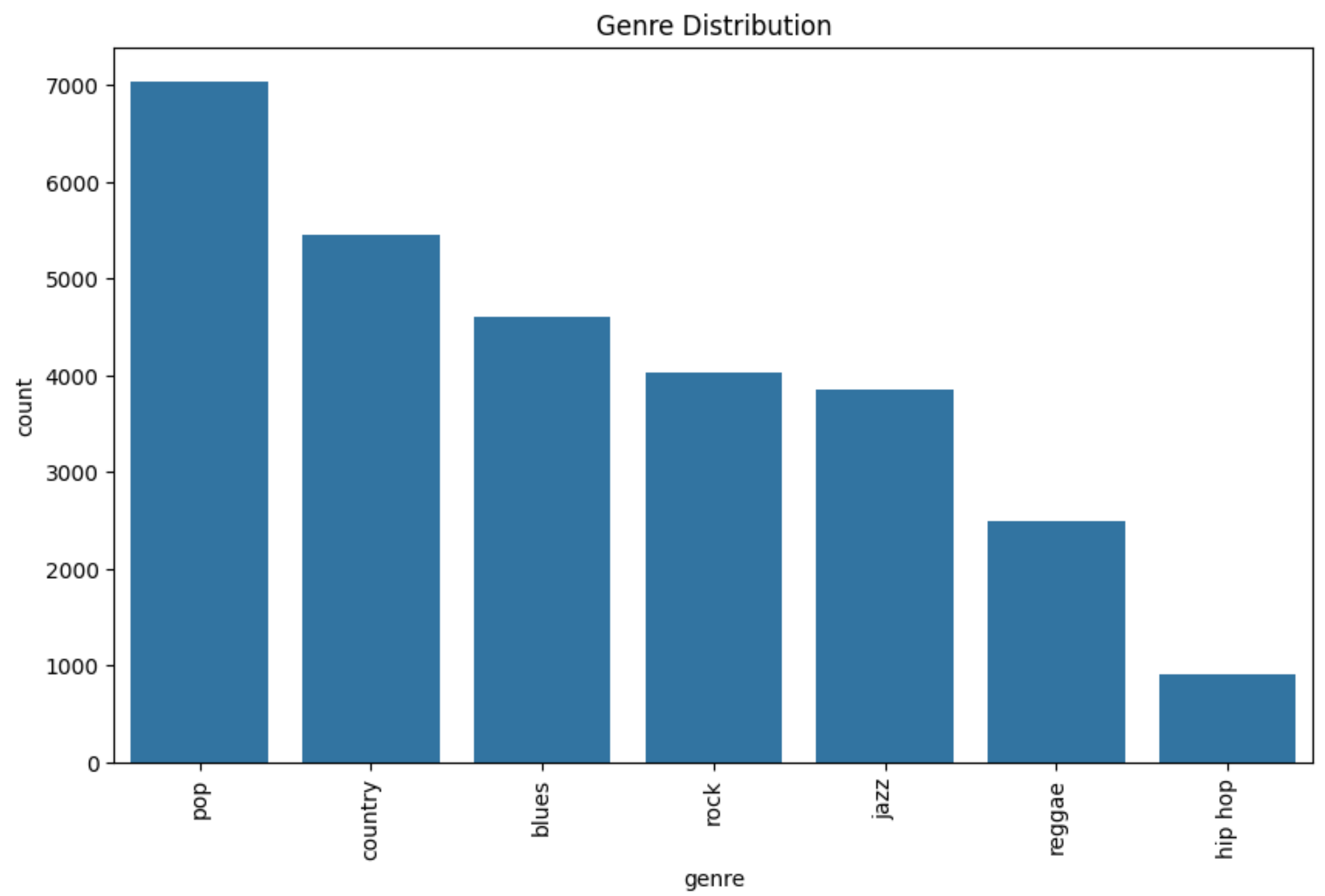
```
artist_name unique values: 5426
artist_name
johnny cash        190
ella fitzgerald    188
dean martin        146
willie nelson      131
george jones       107
Name: count, dtype: int64
track_name unique values: 23689
track_name
tonight        17
stay           15
hold on        15
without you    14
goodbye        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
```

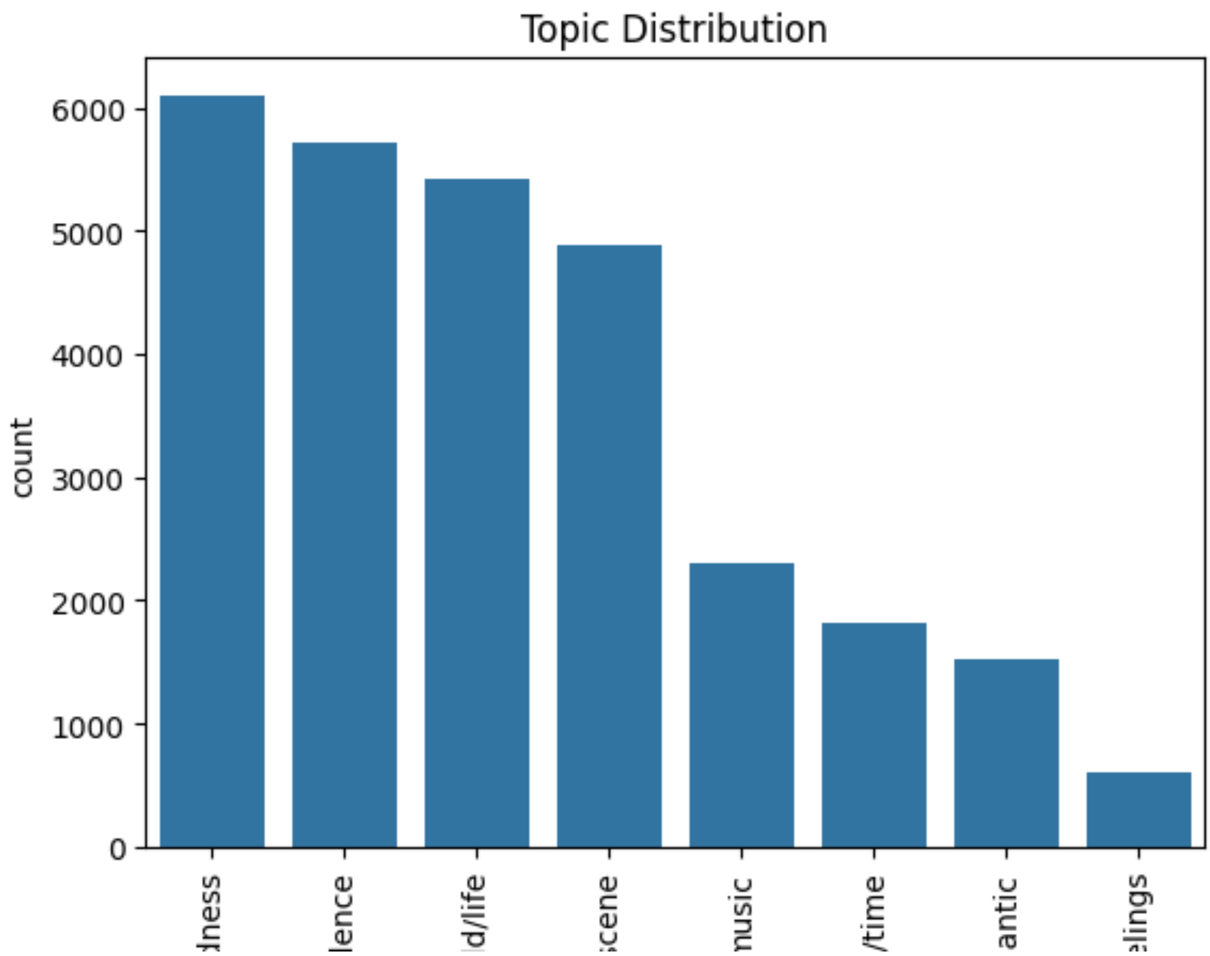
```
import matplotlib.pyplot as plt
import seaborn as sns
# Histogram for numerical columns
df[numerical_cols].hist(bins=20, figsize=(15, 10))
plt.tight_layout()
plt.show()
```

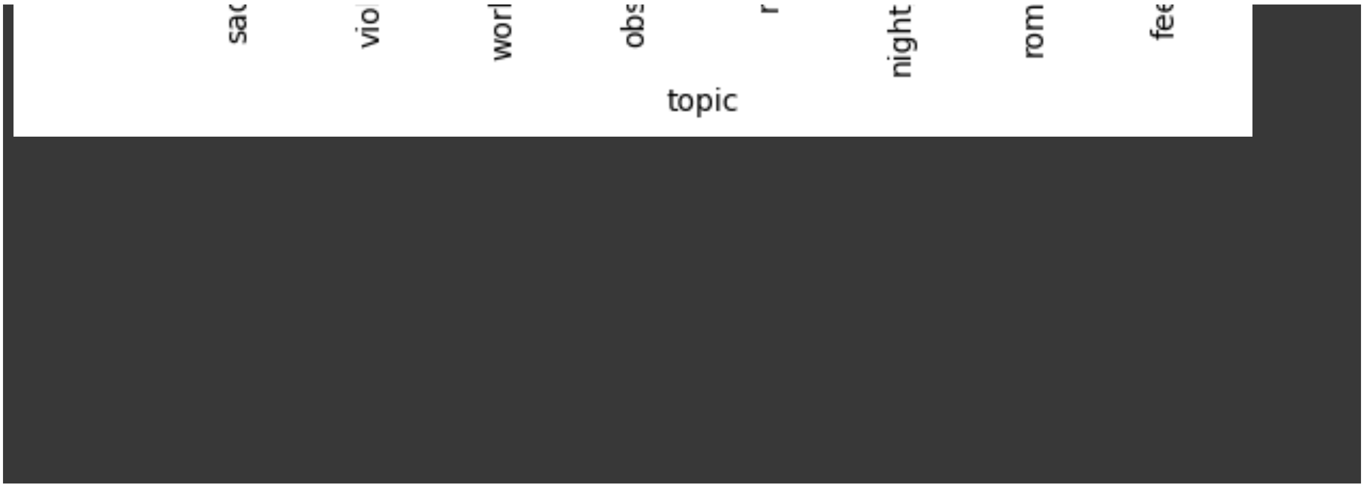



```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='genre', order=df['genre'].value_counts().index)
plt.title('Genre Distribution')
plt.xticks(rotation=90)
plt.show()
# Plot for other categorical columns if needed (e.g., artist_name, topic)
sns.countplot(data=df, x='topic', order=df['topic'].value_counts().index)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

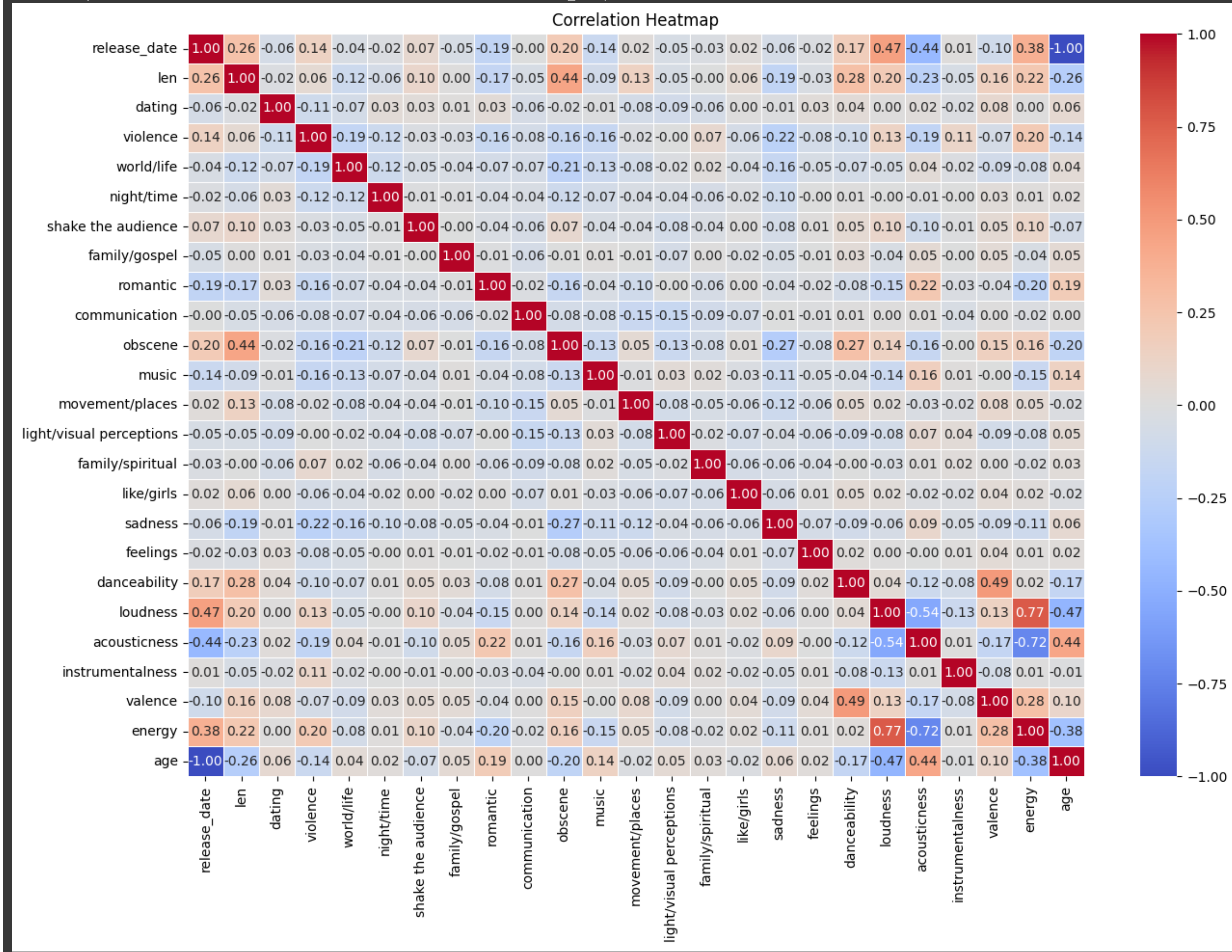

Topic Distribution
count
6000
5000
4000
3000
2000
1000
0



Data Preprocessing

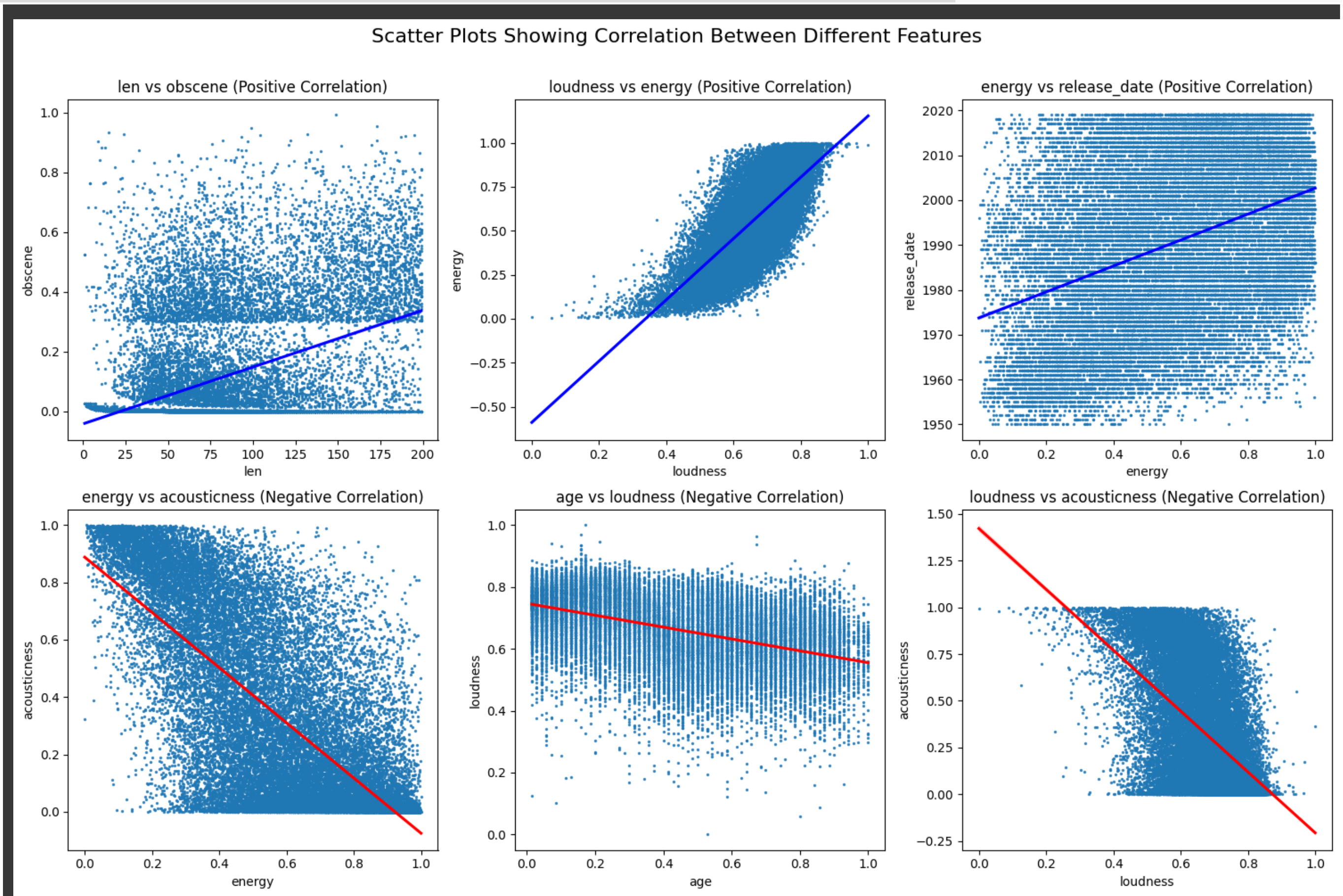
```
# Check for missing values
missing_data = df.isnull().sum()
missing_data_percentage = (missing_data / len(df)) * 100
print(missing_data[missing_data > 0])
print("Missing Data Percentage:\n", missing_data_percentage[missing_data_per

# 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')
```

```
Series([], dtype: int64)
Missing Data Percentage:
 Series([], dtype: float64)
Text(0.5, 1.0, 'Correlation Heatmap')
```




```
# Define the pairs for plotting
positive_pairs = [('len', 'obscene'), ('loudness', 'energy'), ('energy', 're
negative_pairs = [('energy', 'acousticness'), ('age', 'loudness'), ('loudnes
# Create a figure and axis for each plot
fig, axs = plt.subplots(2, 3, figsize=(15, 10)) # 2 rows, 3 columns for 6 pl
fig.suptitle('Scatter Plots Showing Correlation Between Different Features',
# Plot positive correlations with line of best fit
```

```python
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': 2},
  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': 2},
  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()
```



```
# Join all lyrics and create a word cloud
all_lyrics = ' '.join(df['lyrics'])
wordcloud = WordCloud(width=800, height=400, background_color='white').gener
# Display the wordcloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

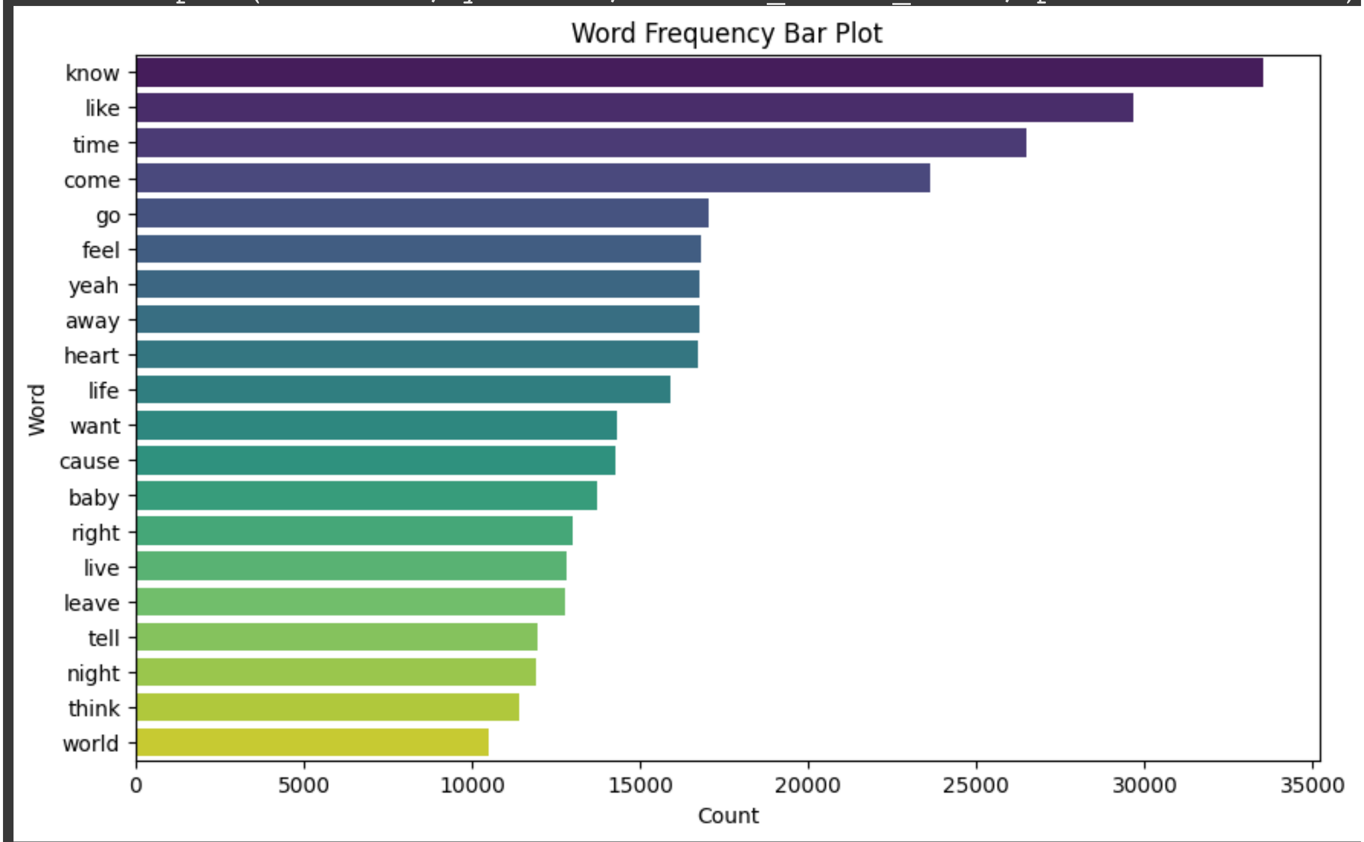

```
from collections import Counter
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

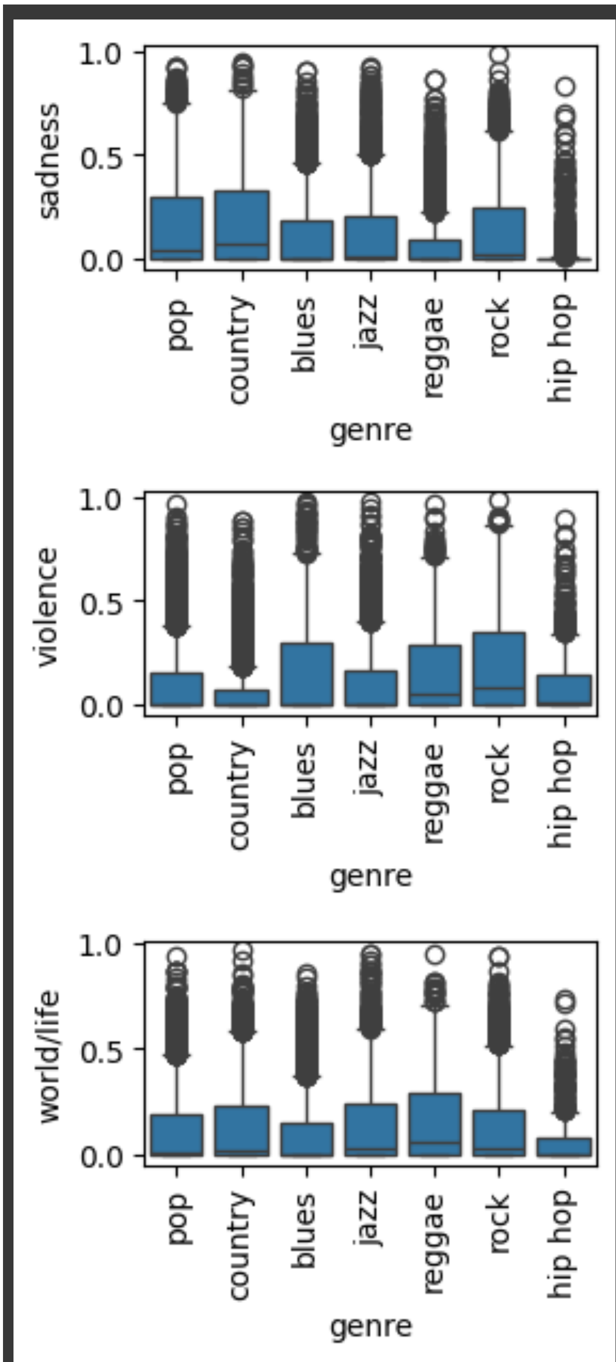
```
# Tokenizing and removing stopwords
tokens = [word for word in all_lyrics.split() if word.lower() not in stop_wc
# Get the most common words
common_words = Counter(tokens).most_common(20)
# Example common_words list
# Convert the list of tuples into a DataFrame
```

```
 7 df_common_words = pd.DataFrame(common_words, columns=['Word', 'Count'])
 8 # Sort by count in descending order for better visualization
 9 df_common_words = df_common_words.sort_values(by='Count', ascending=False)
10 # Create the bar plot using seaborn
11 plt.figure(figsize=(10, 6))
12 sns.barplot(x='Count', y='Word', data=df_common_words, palette='viridis')
13 # Set plot labels and title
14 plt.title('Word Frequency Bar Plot')
15 plt.xlabel('Count')
16 plt.ylabel('Word')
17 # Display the plot
18 plt.show()
```

```
<ipython-input-23-db81834be142>:12: FutureWarning:

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

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

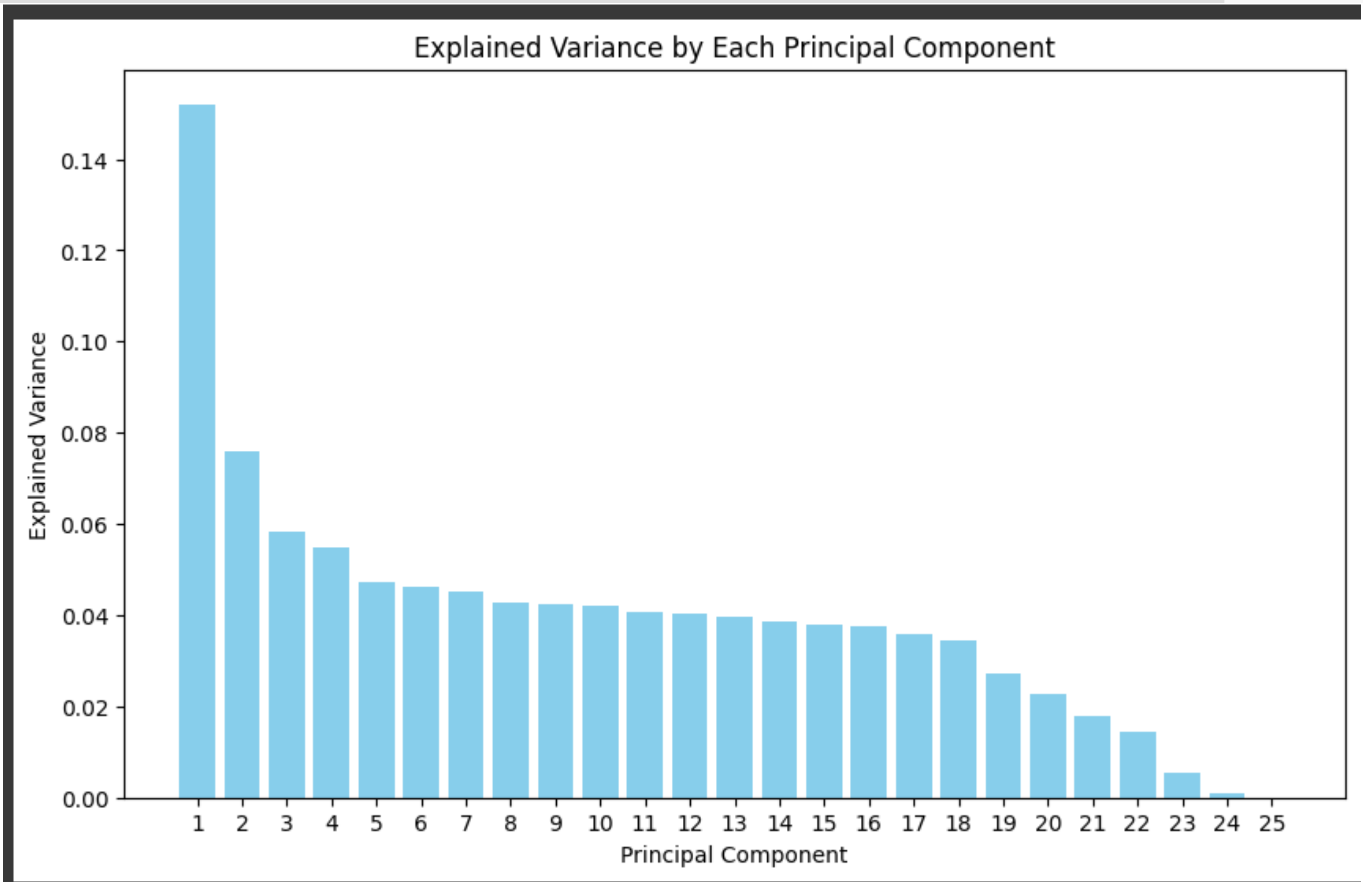


```
# Boxplot for numeric features based on genre
plt.figure(figsize=(6, 3))
plt.subplot(2,2,1)
sns.boxplot(data=df, x='genre', y='sadness') # Example: sadness by genre
plt.xticks(rotation=90)
plt.show()

# Boxplot for numeric features based on genre
plt.figure(figsize=(6, 3))
plt.subplot(2,2,2)
sns.boxplot(data=df, x='genre', y='violence') # Example: violence by genre
plt.xticks(rotation=90)
plt.show()

# Boxplot for numeric features based on genre
plt.figure(figsize=(6, 3))
plt.subplot(2,2,3)
sns.boxplot(data=df, x='genre', y='world/life')
plt.xticks(rotation=90)
plt.show()
```

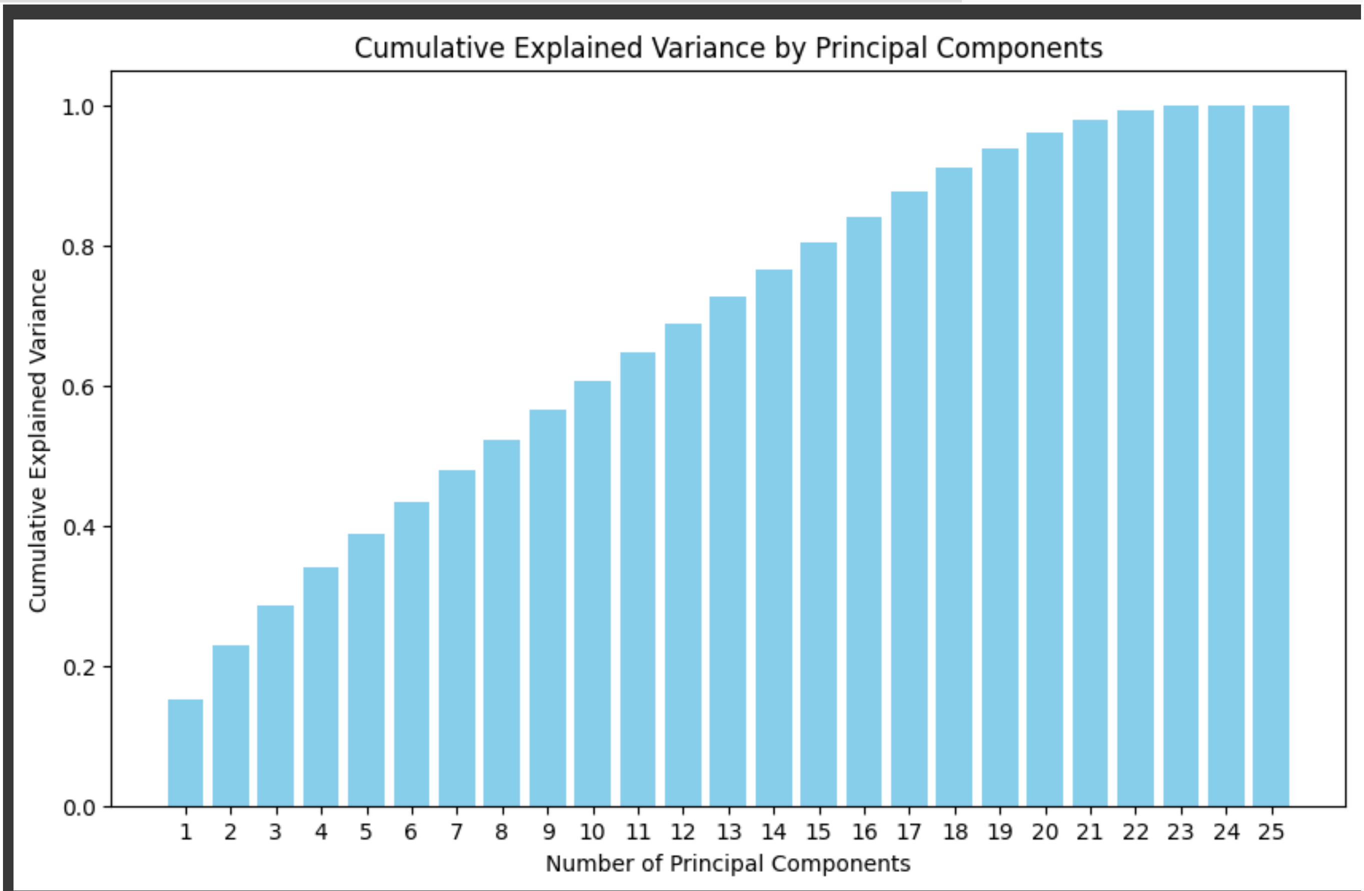



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
# Assuming your DataFrame is called 'df'
# Select numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
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='sk
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 = 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_explair
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()
```




```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
# Assuming your DataFrame is called 'df'
# 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}:")
  print(components_df.iloc[i])
  print("\n")
```

```
First 3 Principal Components:
Principal Component 1:
release_date                0.394588
len                         0.254156
dating                     -0.020535
violence                    0.123877
world/life                 -0.062965
night/time                 -0.009484
shake the audience          0.090472
family/gospel              -0.026792
romantic                   -0.170017
communication              -0.011355
obscene                     0.210077
music                      -0.121121
movement/places             0.055093
light/visual perceptions   -0.070238
family/spiritual           -0.016020
like/girls                  0.026608
sadness                    -0.101302
feelings                   -0.005960
danceability                0.152552
loudness                    0.381933
acousticness               -0.384740
instrumentalness           -0.021810
valence                     0.115731
energy                      0.396450
age                        -0.394588
Name: 0, dtype: float64


Principal Component 2:
release_date               -0.213723
len                         0.281174
dating                      0.139691
violence                   -0.259469
world/life                 -0.139391
```

```
night/time                 0.032565
shake the audience         0.091527
family/gospel              0.116300
romantic                   0.020838
communication             -0.030348
obscene                    0.363784
music                      0.073331
movement/places            0.145732
light/visual perceptions  -0.156542
family/spiritual          -0.050574
like/girls                 0.094260
sadness                   -0.156949
feelings                   0.044765
danceability               0.461304
loudness                  -0.122559
acousticness               0.084466
instrumentalness          -0.123907
valence                    0.469775
energy                    -0.080393
age                        0.213723
Name: 1, dtype: float64
```

```python
# Step 1: 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: Load the dataset
file_path = "/content/tcc_ceds_music.csv"  # Path to the uploaded file in Cc
df = pd.read_csv(file_path)

# Step 3: Preprocessing
# Drop unnecessary columns
if 'Unnamed: 0' in df.columns:
    df.drop(columns=['Unnamed: 0'], inplace=True)

# Numerical Feature Scaling
numeric_features = ['danceability', 'energy', 'acousticness', 'valence', 'lc
X_numeric = df[numeric_features]

# Scale the numerical features
scaler = StandardScaler()
X_numeric_scaled = scaler.fit_transform(X_numeric)

# Text Feature Extraction (TF-IDF + SVD)
```

```
# Preprocess lyrics: lowercase and keep alphabetic tokens only
df['processed_lyrics'] = df['lyrics'].apply(
    lambda x: ' '.join([word.lower() for word in str(x).split() if word.isal
)

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
X_text = vectorizer.fit_transform(df['processed_lyrics'])

# Reduce Dimensionality using SVD
svd = TruncatedSVD(n_components=100, random_state=42)
X_text_reduced = svd.fit_transform(X_text)

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

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

# Function to recommend songs
def recommend_songs(song_name, df, knn_model, combined_features):
    # Check if song exists
    if song_name not in df['track_name'].values:
        print(f"Error: The song '{song_name}' does not exist in the dataset.
        return None

    # Find the index of the input song
    song_index = df[df['track_name'] == song_name].index[0]

    # Get the feature vector for the input song
    song_features = combined_features[song_index].reshape(1, -1)

    # Find the nearest neighbors
    distances, indices = knn_model.kneighbors(song_features)

    # Get the indices of the top 10 most similar songs (excluding the input
    similar_song_indices = indices.flatten()[1:]

    # Return the top 10 most similar songs
    return df.iloc[similar_song_indices]

# Step 6: Test the System
# Input a song name and get recommendations
input_song = "i believe"  # Replace with any song name from the dataset
recommendations = recommend_songs(input_song, df, knn, X_combined)

# Display recommendations
```

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

```
                                              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
17595  when you're smiling (the whole world smiles wi...      nat king cole
94                          september in the rain       the platters
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
17595      0.392397  0.340320      0.860442  0.342539
94         0.286256  0.235211      0.969879  0.241550
152        0.375068  0.287265      0.894578  0.389942
```

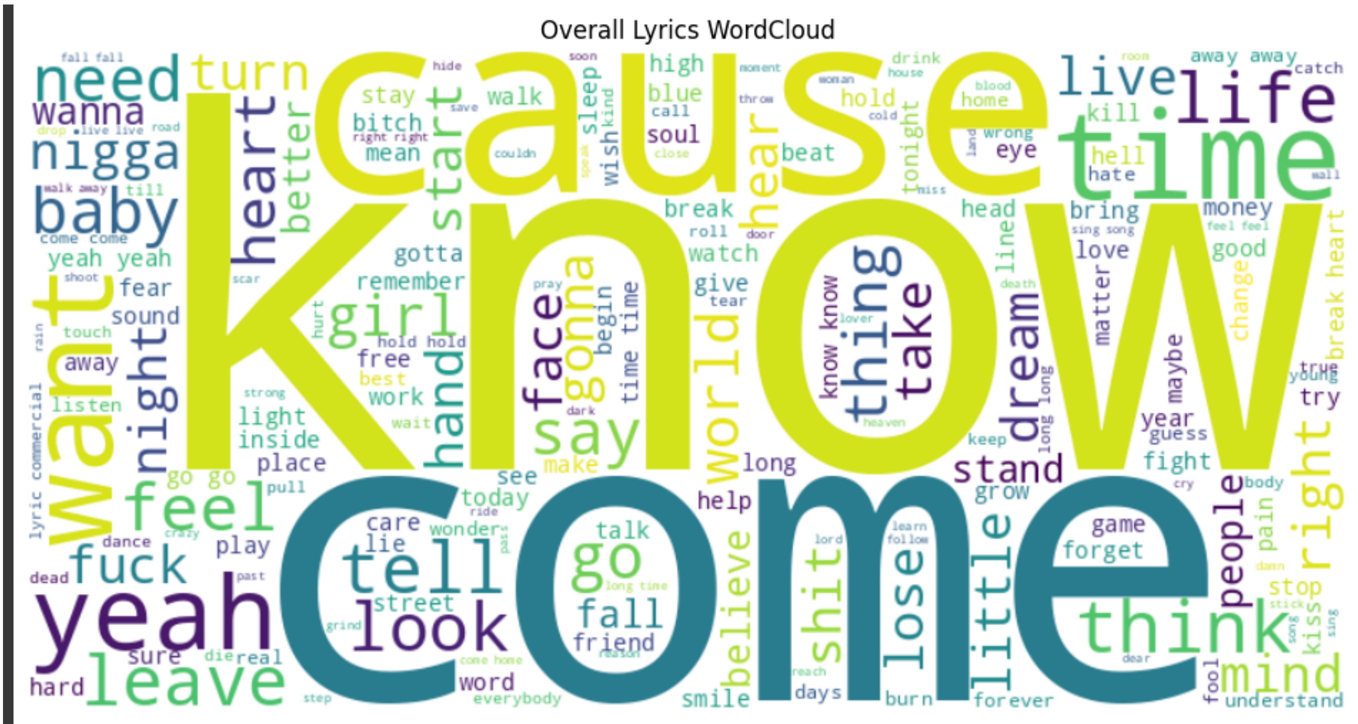
```
# Step 1: Check if the input song exists in the dataset
input_song = "i believe" # Replace with any song name from the dataset
# Check if the song exists in the dataset
if input_song not in df['track_name'].values:
  print(f"Error: The song '{input_song}' does not exist in the dataset.")
else:
  # Step 2: Find the index of the input song
  song_index = df[df['track_name'] == input_song].index[0]

  # Step 3: Get the feature vector for the input song
  song_features = X_combined[song_index].reshape(1, -1)

  # Step 4: Find the nearest neighbors
  distances, indices = knn.kneighbors(song_features)

  # Step 5: Get the indices of the top 10 most similar songs (excluding the
  similar_song_indices = indices.flatten()[1:]

  # Step 6: Return the top 10 most similar songs
  recommendations = df.iloc[similar_song_indices]
  print(recommendations[['track_name', 'artist_name', 'danceability', 'energ
```

```
                                           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
17595  when you're smiling (the whole world smiles wi...     nat king cole
94                                september in the rain     the platters
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
17595      0.392397  0.340320      0.860442  0.342539
94         0.286256  0.235211      0.969879  0.241550
152        0.375068  0.287265      0.894578  0.389942
```

Recomender system, Using NLP

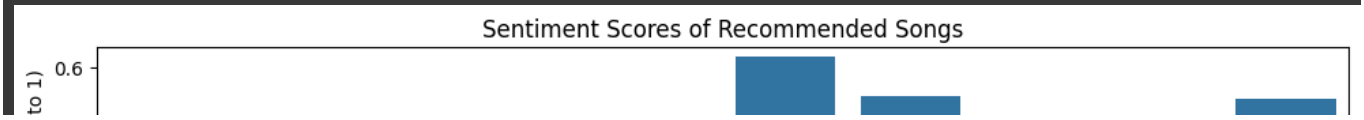
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import StandardScaler
from textblob import TextBlob
import re
import string

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

# 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.po

# 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))
```

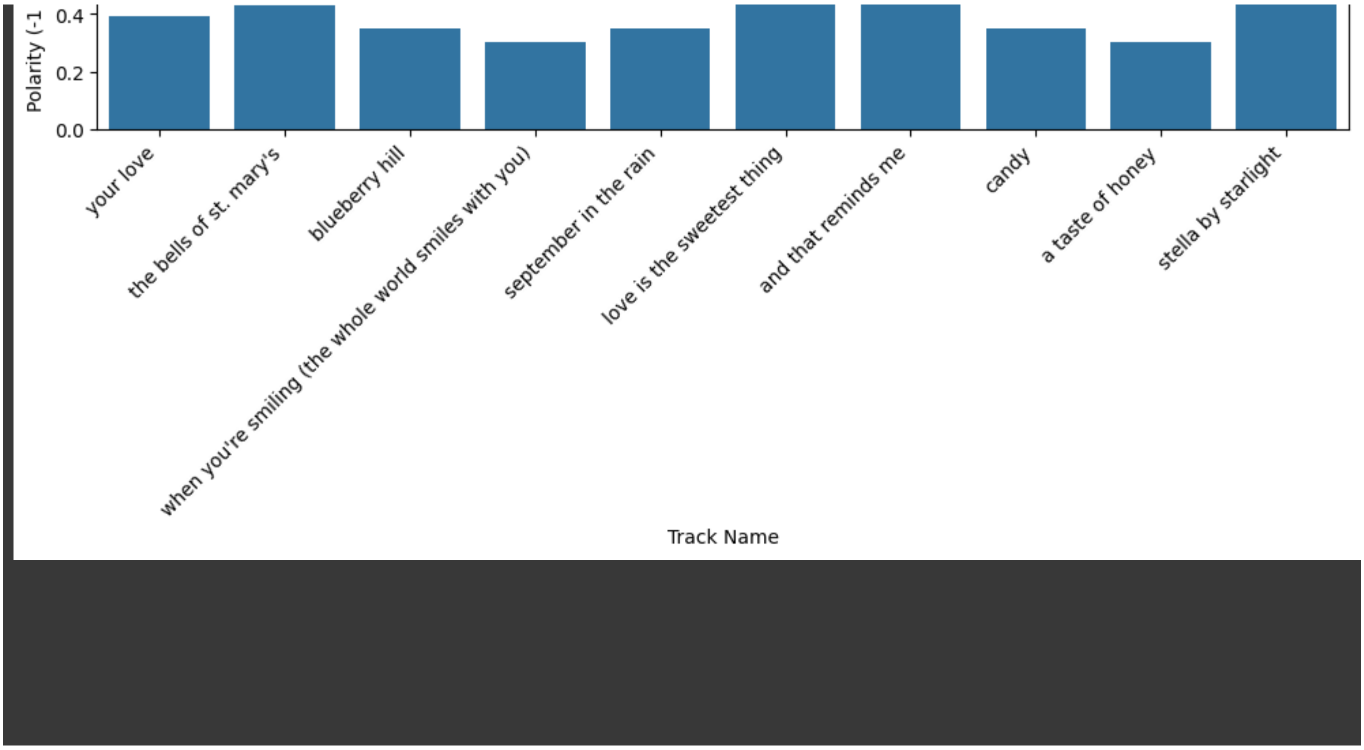
```
X_combined = np.hstack((X_numeric_scaled, X_text_reduced))

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

# Optional: Show WordCloud for general lyrics
all_lyrics = ' '.join(df['clean_lyrics'])
wordcloud = WordCloud(width=800, height=400, background_color='white').genera

plt.figure(figsize=(12, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Overall Lyrics WordCloud')
plt.show()

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

    song_index = df[df['track_name'] == song_name].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 itself

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

# Step 9: Test the recommender
input_song = "i believe"  # Replace with any song from your dataset
recommendations = recommend_songs(input_song, df, knn, X_combined)

if recommendations is not None:
    print("Top 10 similar songs to:", input_song)
    display(recommendations[['track_name', 'artist_name', 'danceability', 'en

    # Step 10: Visualize sentiment of recommended songs
    plt.figure(figsize=(10, 5))
    sns.barplot(x='track_name', 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('Track Name')
    plt.tight_layout()
    plt.show()
```



Top 10 similar songs to: i believe

	track_name	artist_name	danceability	energy	valence	sentiment
12900	your love	carla thomas	0.384815	0.310289	0.317807	0.389603
47	the bells of st. mary's	the drifters	0.281924	0.205180	0.231245	0.430000
542	blueberry hill	the lettermen	0.323080	0.366347	0.331204	0.350000
17595	when you're smiling (the whole world smiles wi...	nat king cole	0.392397	0.340320	0.342539	0.300000
94	september in the rain	the platters	0.286256	0.235211	0.241550	0.350000
17121	love is the sweetest thing	ray noble	0.284090	0.309288	0.322960	0.637500
17388	and that reminds me	della reese	0.358822	0.241218	0.309563	0.500000
12742	candy	big maybelle	0.365320	0.240217	0.341509	0.350000
17658	a taste of honey	tony bennett	0.356655	0.237213	0.348722	0.300000
17322	stella by starlight	miles davis	0.425972	0.182157	0.234336	0.491667





Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.